Estruturas Multidimensionais
Árvore R

Prof.a. Bárbara Quintela
Prof. Jose J. Camata
Prof. Marcelo Caniato

barbara@ice.ufjf.br
camata@ice.ufjf.br
marcelo.caniato@ice.ufjf.br

Introdução

- Objetos de dados espaciais geralmente cobrem áreas de espaços multidimensionais e não são bem representados por localizações de pontos (quadtrees).
- Árvore-R (R-Tree) é uma estrutura de dados em árvore usada para armazenar índices de dados espaciais multidimensionais de maneira eficiente.

Introdução

- As árvore-R são estruturas espaciais em que a intersecção entre objetos é permitida
- Foram propostas por em 1984 por Antonin Guttman
  - como uma melhoria nas árvores B, com o objetivo de tratar dados geométricos em espaços de maior dimensão

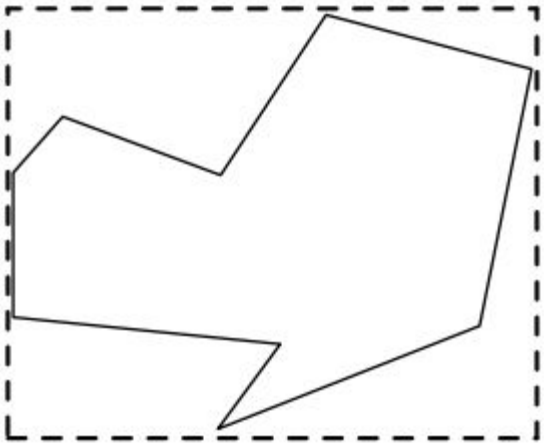
Aplicação:

- Alguns exemplos de aplicação são mencionados abaixo:
  - Indexação de informações multidimensionais.
  - Manipulação de coordenadas geoespaciais.
  - Implementação de mapas virtuais.
  - Tratamento de dados de jogos.

Representação:

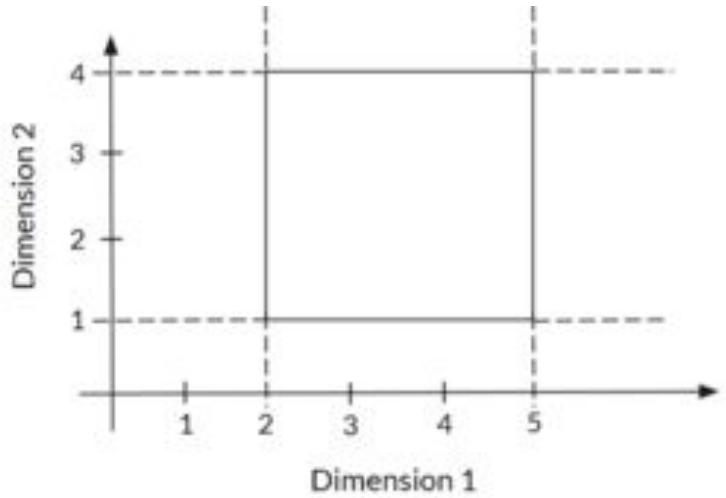
- Seu funcionamento básico consiste em definir “retângulos” que contenham os dados inseridos na árvore.
- Esses retângulos são chamados MBRs, de *minimum bounding d-dimensional rectangles*

MBR é formado a partir da observação dos limites geométricos mínimo e máximo do contorno do objeto



Árvores-R: Regras Gerais

- Os retângulos n-dimensionais são delimitados por **n** intervalos fechados
- Um retângulo bidimensional, tem dois intervalos
  - Por exemplo: [2,5][1,4]



Árvores-R: Regras Gerais

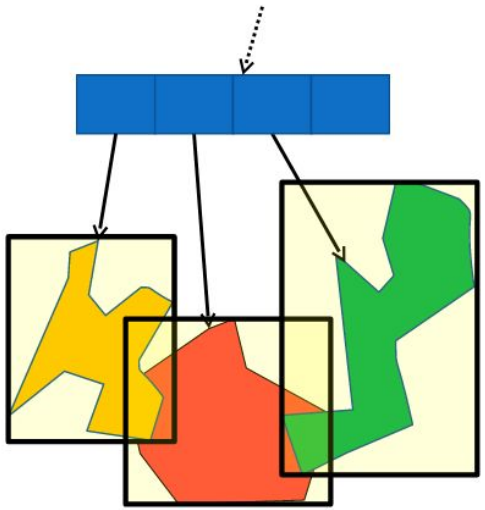
- Árvore-R possui parâmetros para determinar a quantidade de chaves (retângulos) que poderão ocorrer em cada bloco de armazenamento, analogamente à árvore-B
- Todas as folhas aparecem sempre no mesmo nível da árvore.
- Nós internos contêm a delimitação de retângulos que englobam todos os retângulos dos nós nos níveis inferiores.

Árvores-R: Regras Gerais

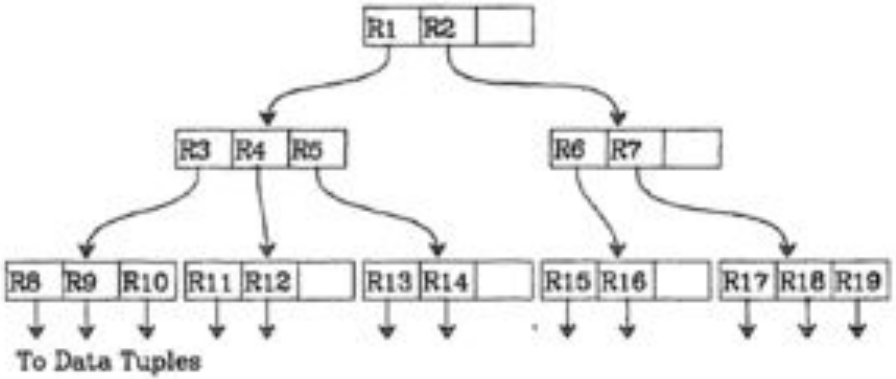
- Árvore-R possui no máximo M entradas em um nó
- O número mínimo de entradas por nó é calculado em função do M
  - m = M/2
  - essa propriedade é utilizada para definir se a árvore precisa ser reorganizada
- Uma árvore-R de ordem (m, M) conterá entre m e M entradas em cada nó da árvore (m < M / 2), exceto a raiz, que terá pelo menos dois nós

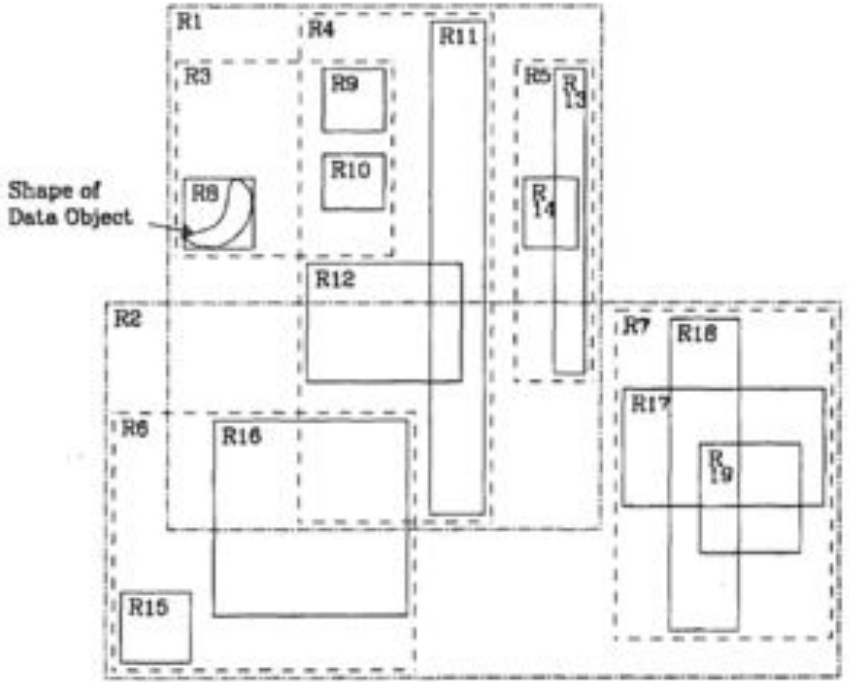
Árvores R

- Importante:
  - Entrada de nós diferentes podem ter sobreposição.



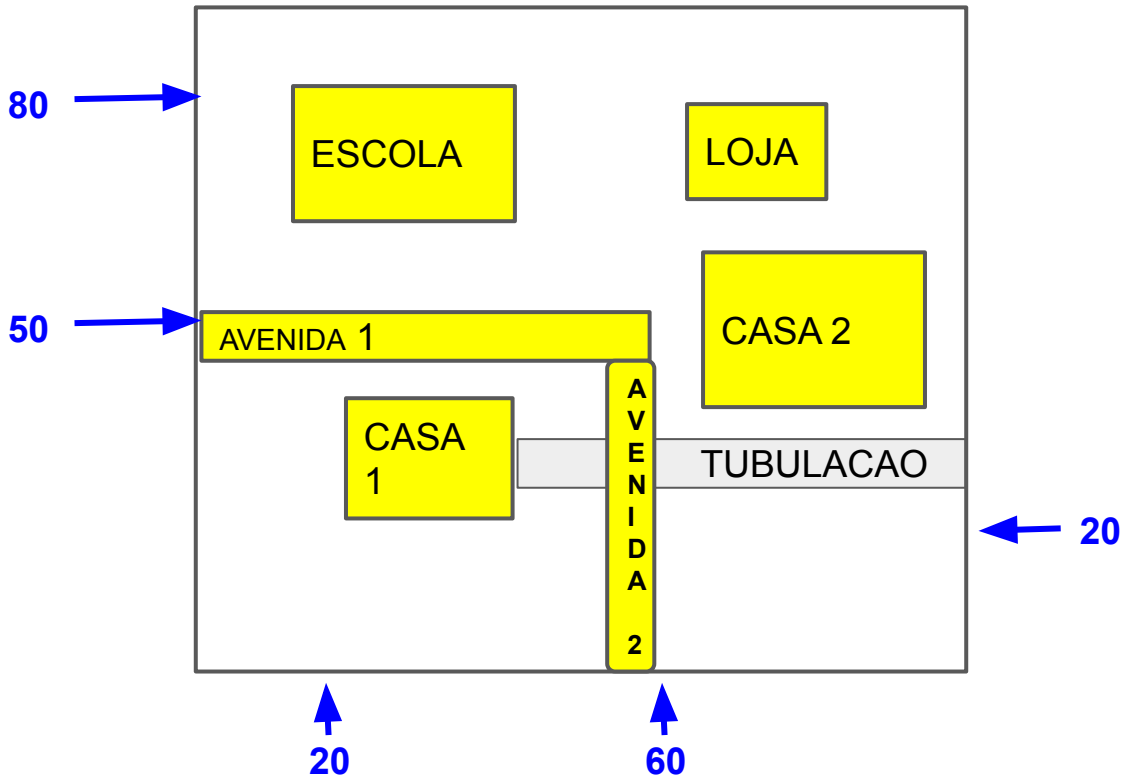
Árvores R


R1 R2
R3 R4 R5
R6 R7
R8 R9 R10
R11 R12
R13 R14
R15 R16
R17 R18 R19
To Data Tuples


Shape of
Data Object
R1
R4
R11
R3
R9
R5
R13
R10
R8
R14
R12
R2
R7
R18
R17
R6
R16
R19
R15

Exemplo - Árvore R (m=2,M=4)

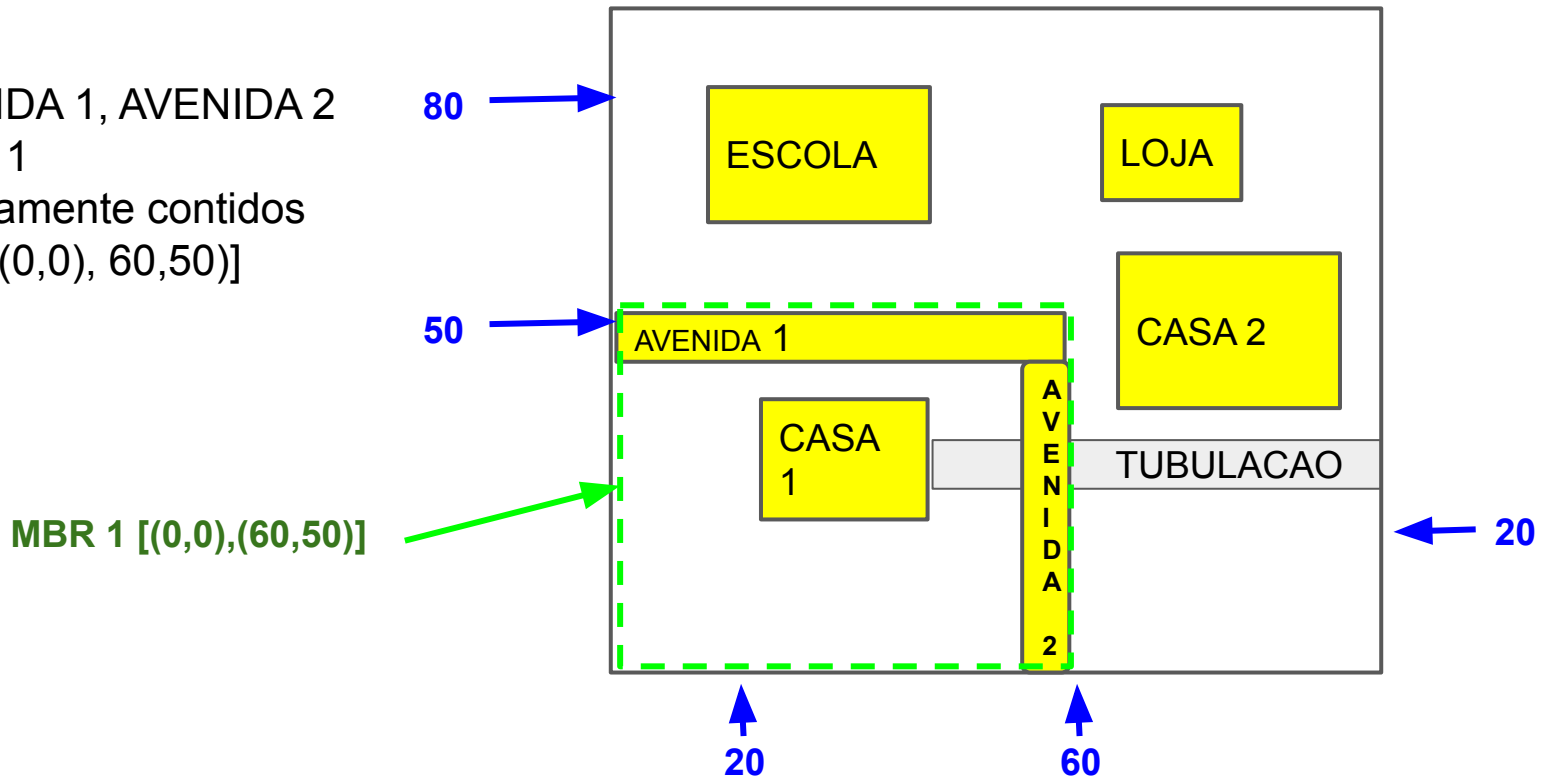
Objetos que queremos representar



Exemplo - Árvore R (m=2,M=4)

Os Objetos
- AVENIDA 1, AVENIDA 2
  CASA 1

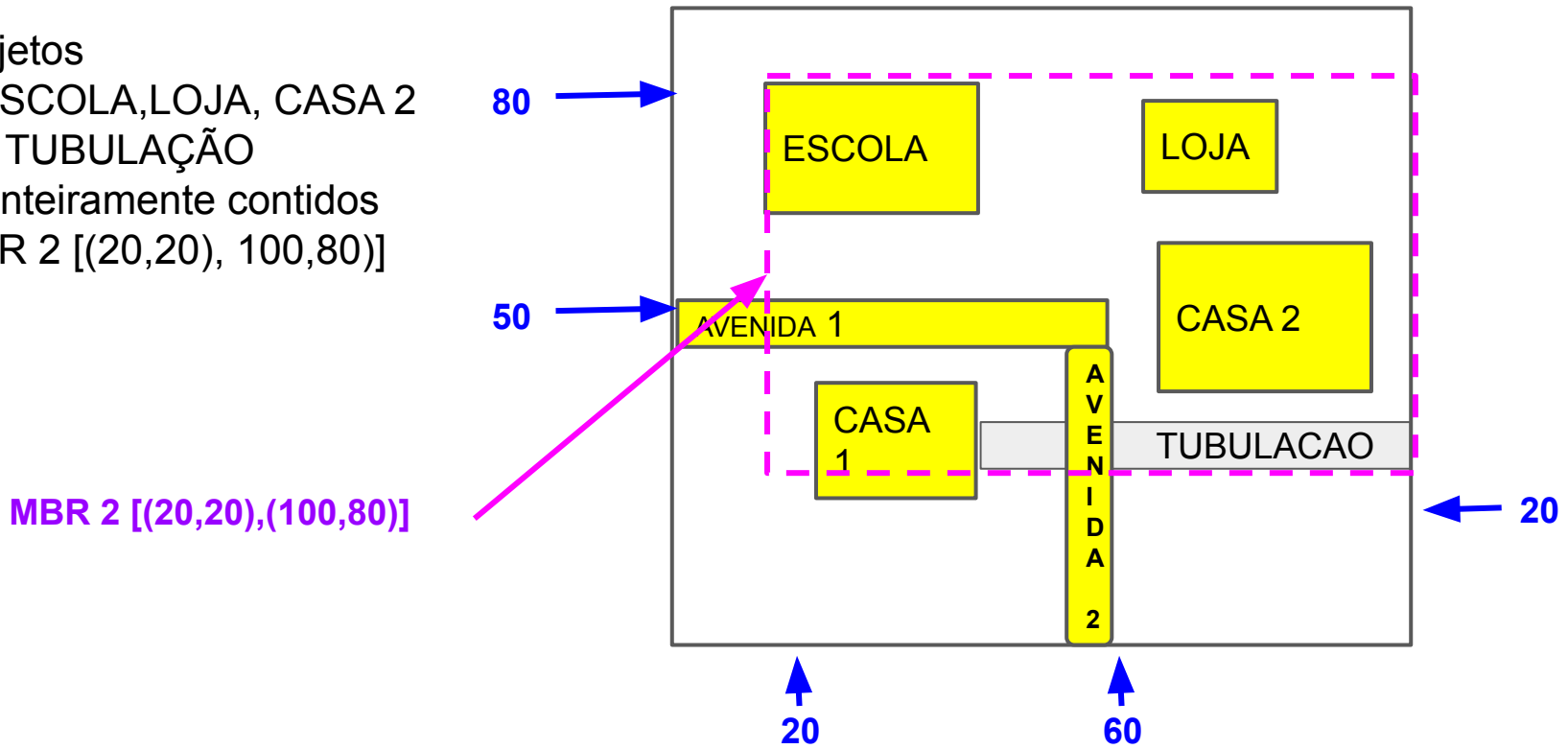
estão inteiramente contidos
no MBR 1 [(0,0), 60,50)]



Exemplo - Árvore R (m=2,M=4)

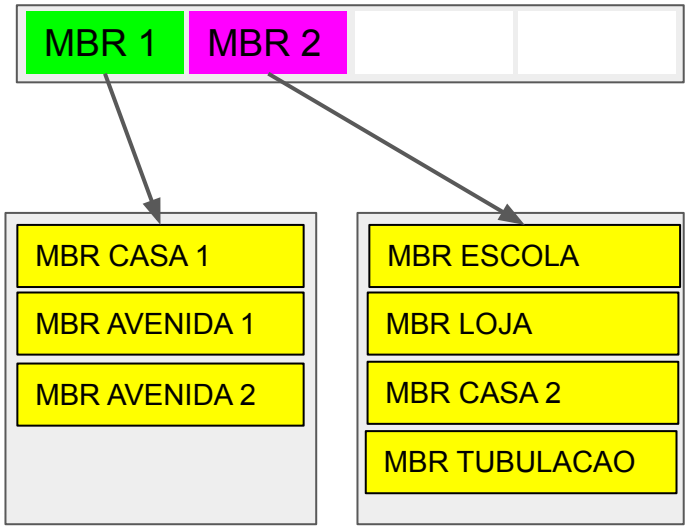
Os Objetos
- ESCOLA,LOJA, CASA 2 e TUBULAÇÃO

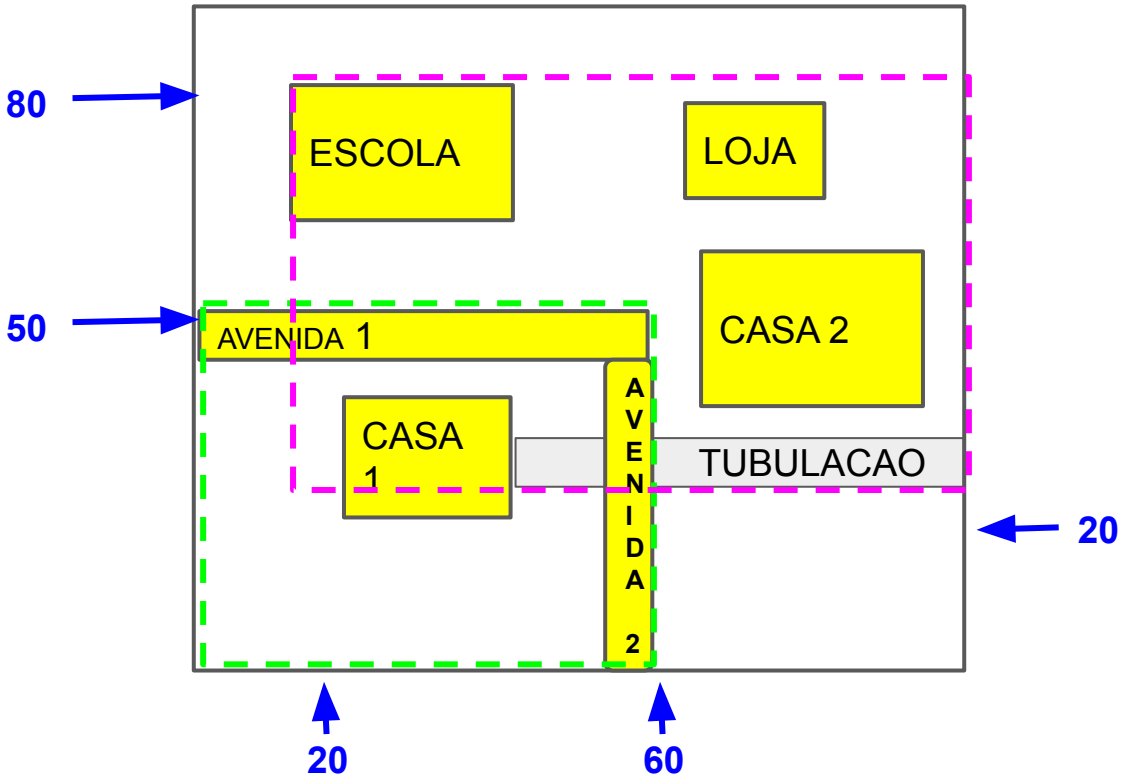
estão inteiramente contidos no MBR 2 [(20,20), 100,80)]



Exemplo - Árvore R (m=2,M=4)

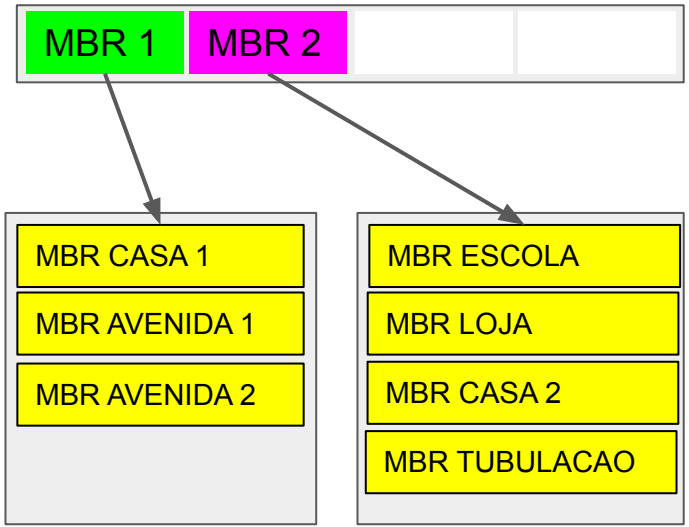
Representação em Árvore R

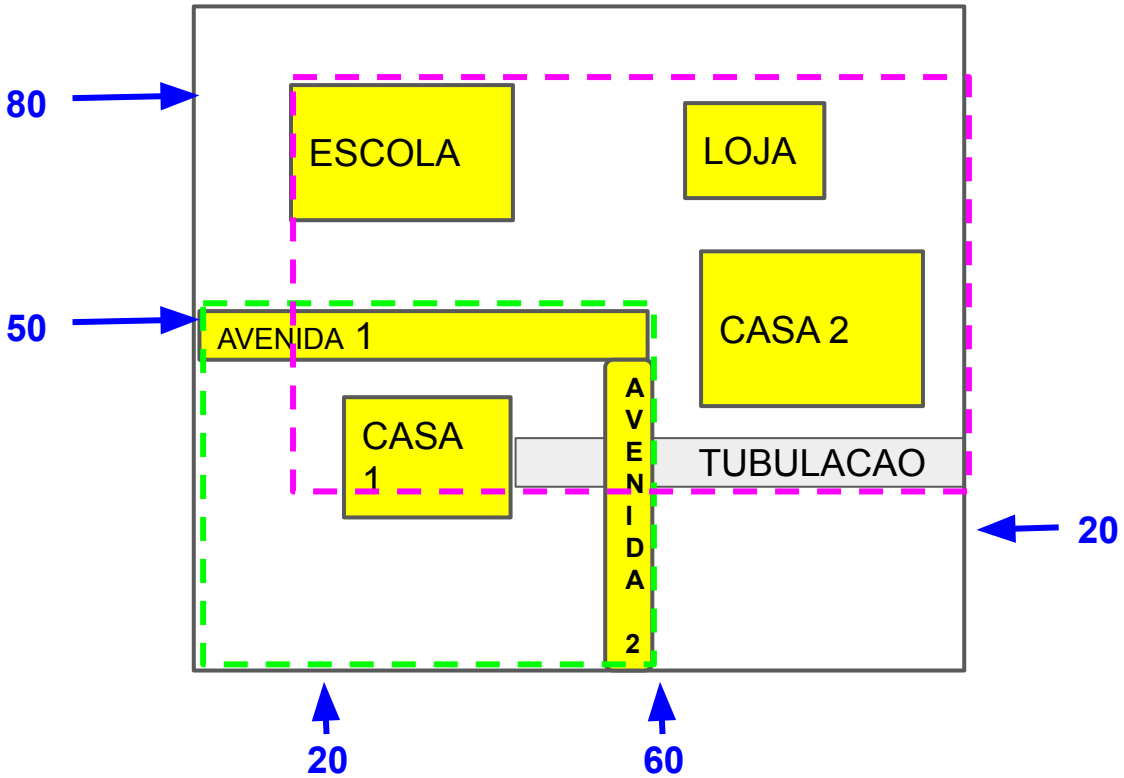




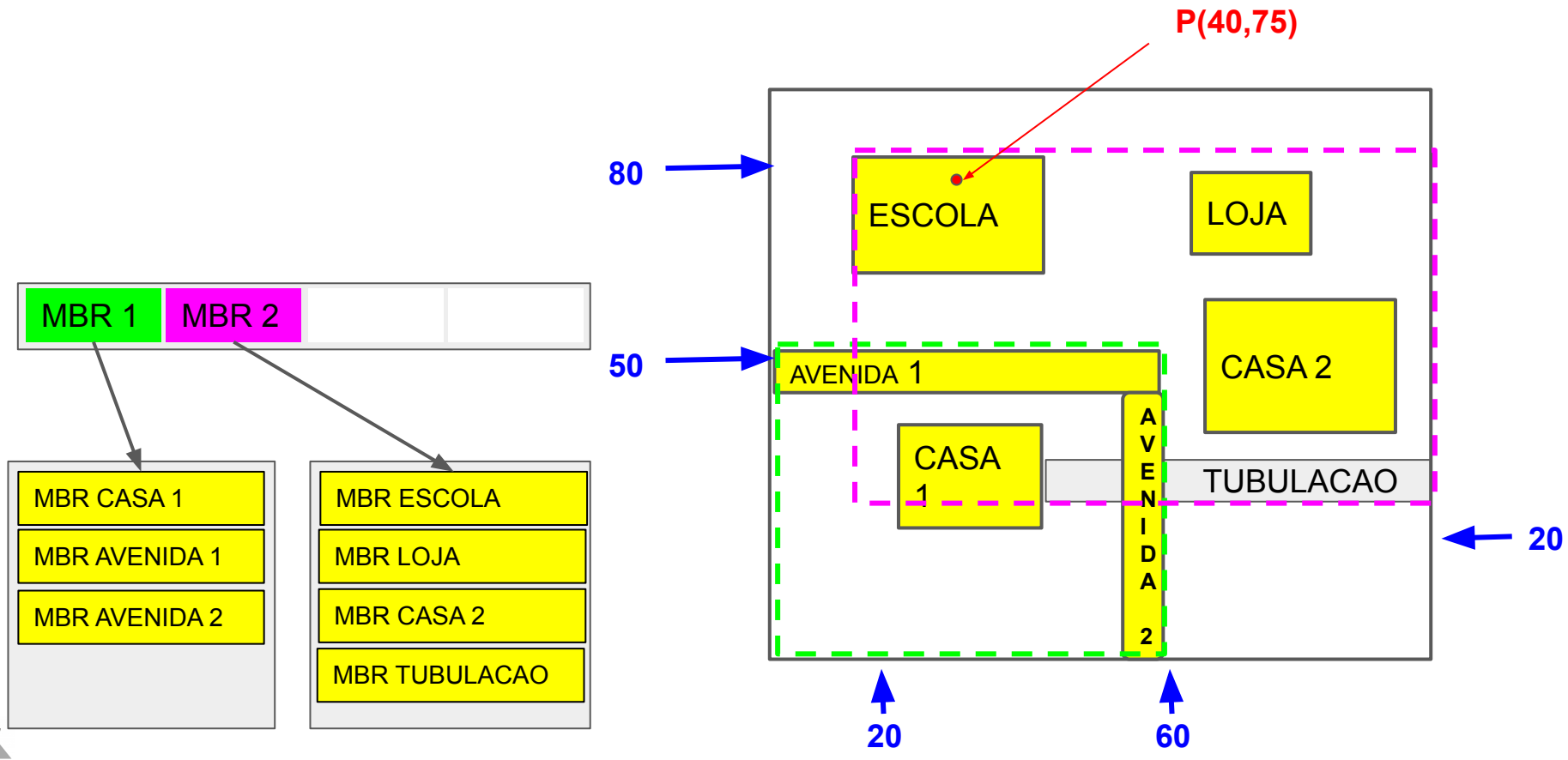
Pesquisa por um Ponto (x,y)

- Algoritmo recursivo
- A pesquisa inicia na raiz da Árvore R



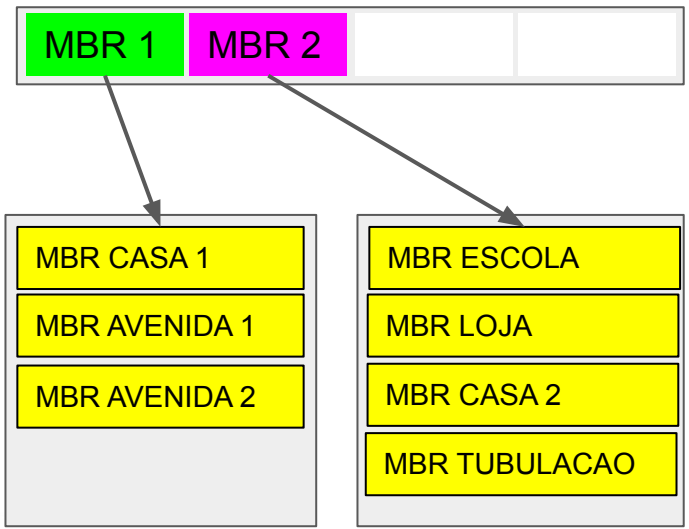


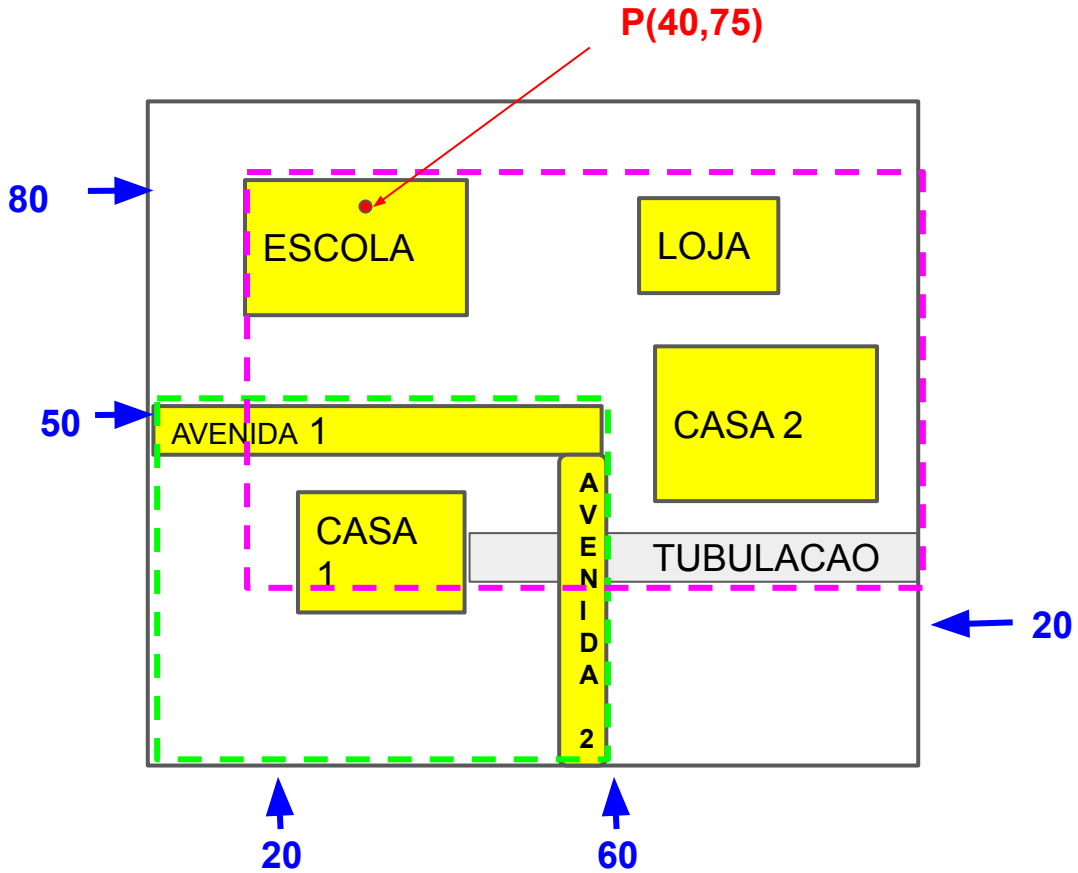
Exemplo: Encontrar objeto que contém o ponto P(40,75)



Exemplo: Encontrar objeto que contém o ponto P(40,75)

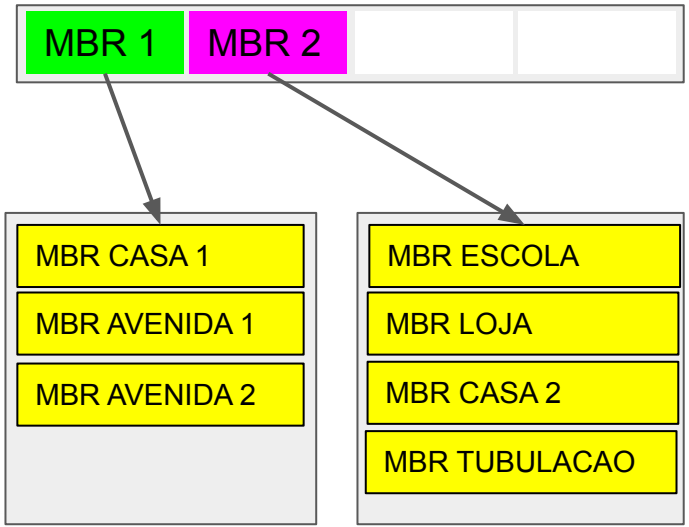
1. P(40,75) está em MBR1 [(0,0),[60,50]]?

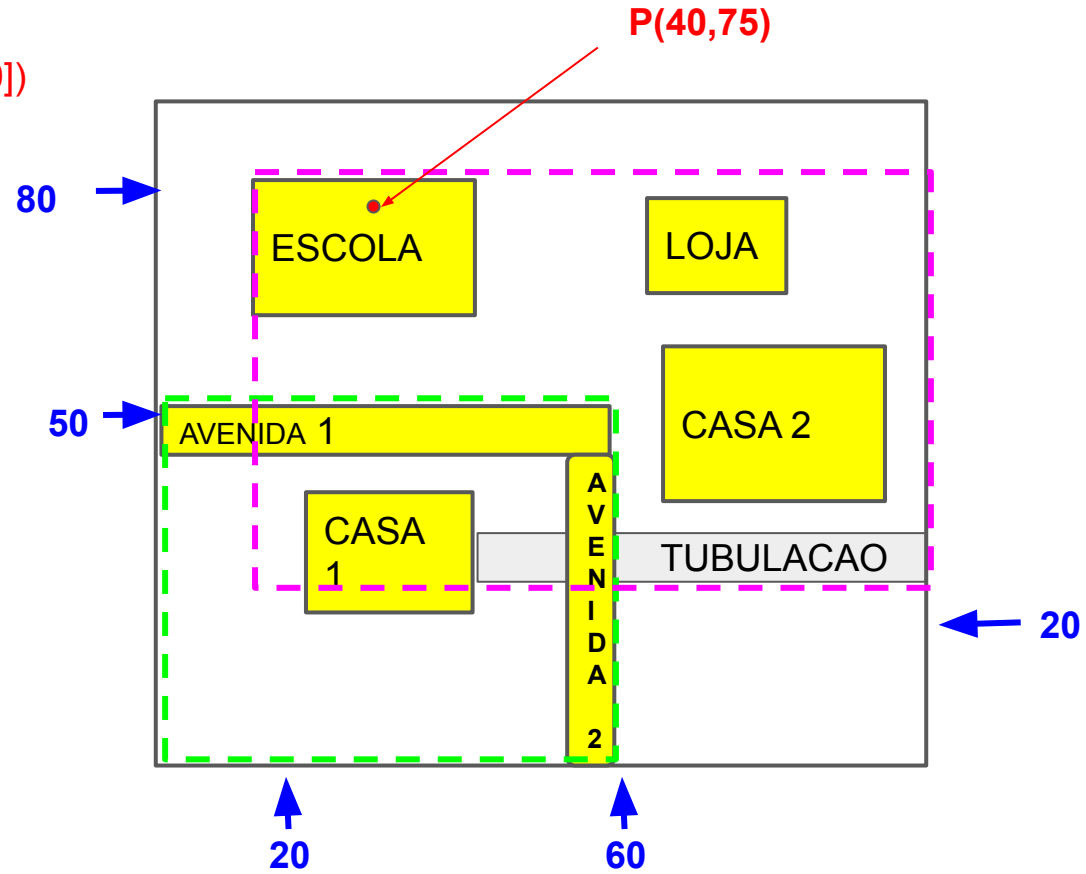




Exemplo: Encontrar objeto que contém o ponto P(40,75)

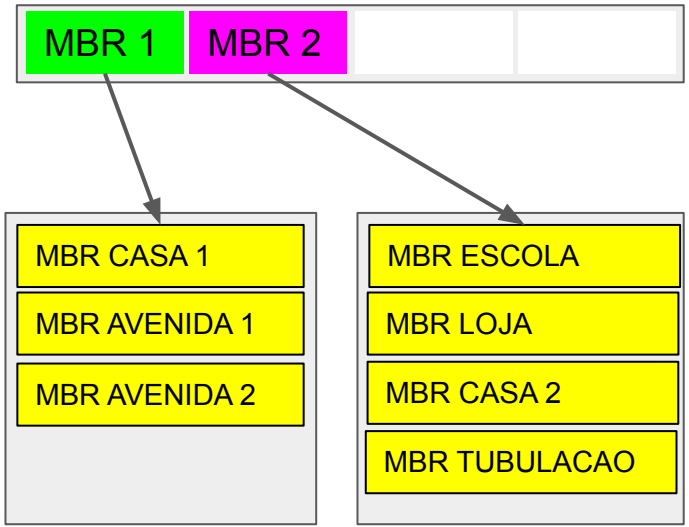
1. P(40,75) **não** está em MBR1 [(0,0),[60,50])

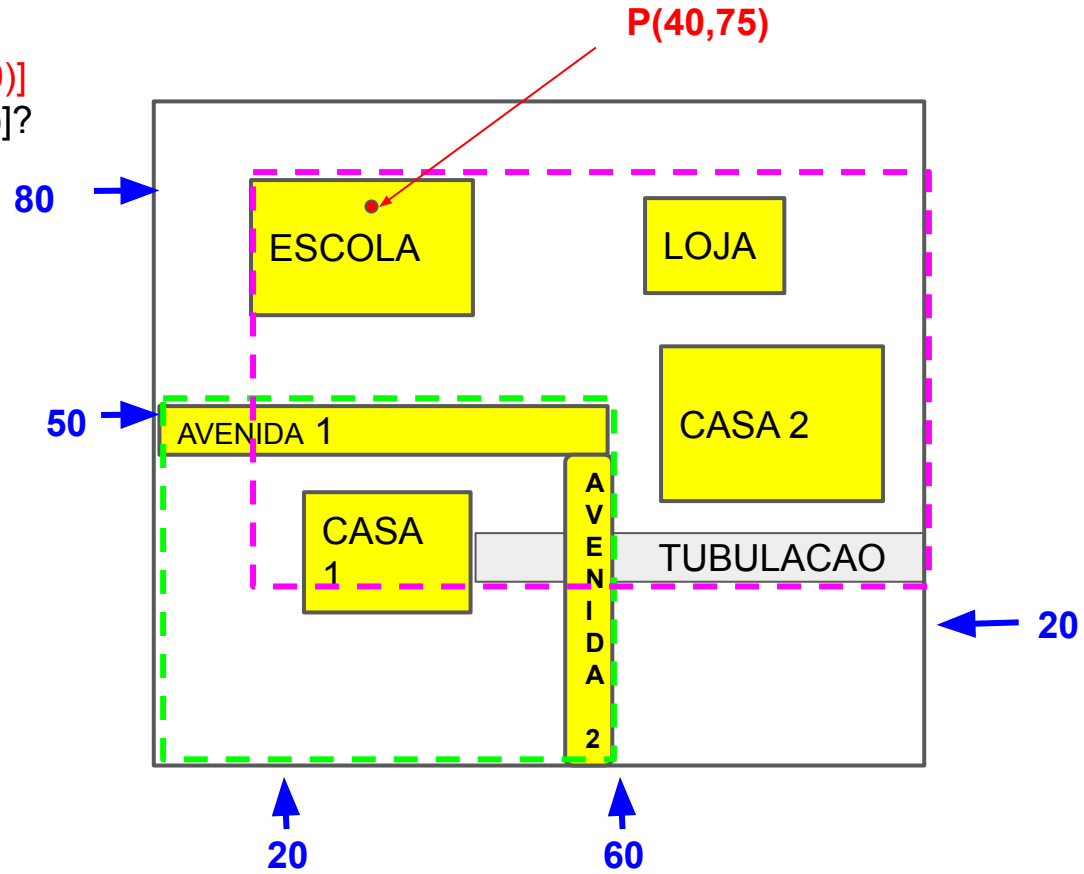




Exemplo: Encontrar objeto que contém o ponto P(40,75)

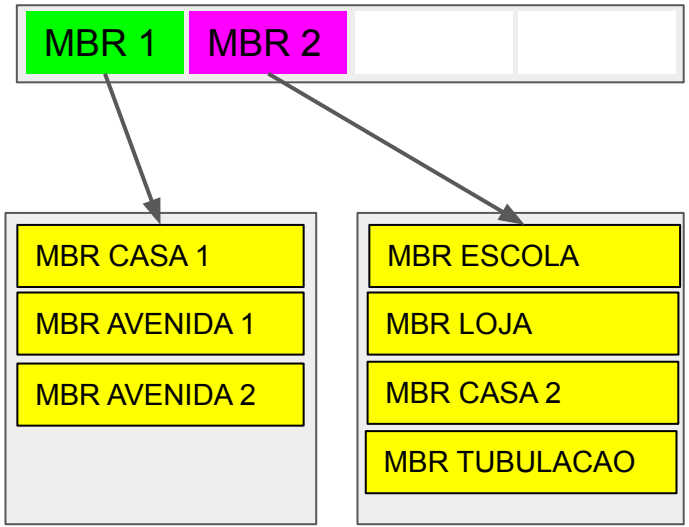
1. P(40,75) **não** está em MBR1 [(0,0),(60,50)]
2. P(40,75) está em MBR2 [(20,20),(20,100)]?

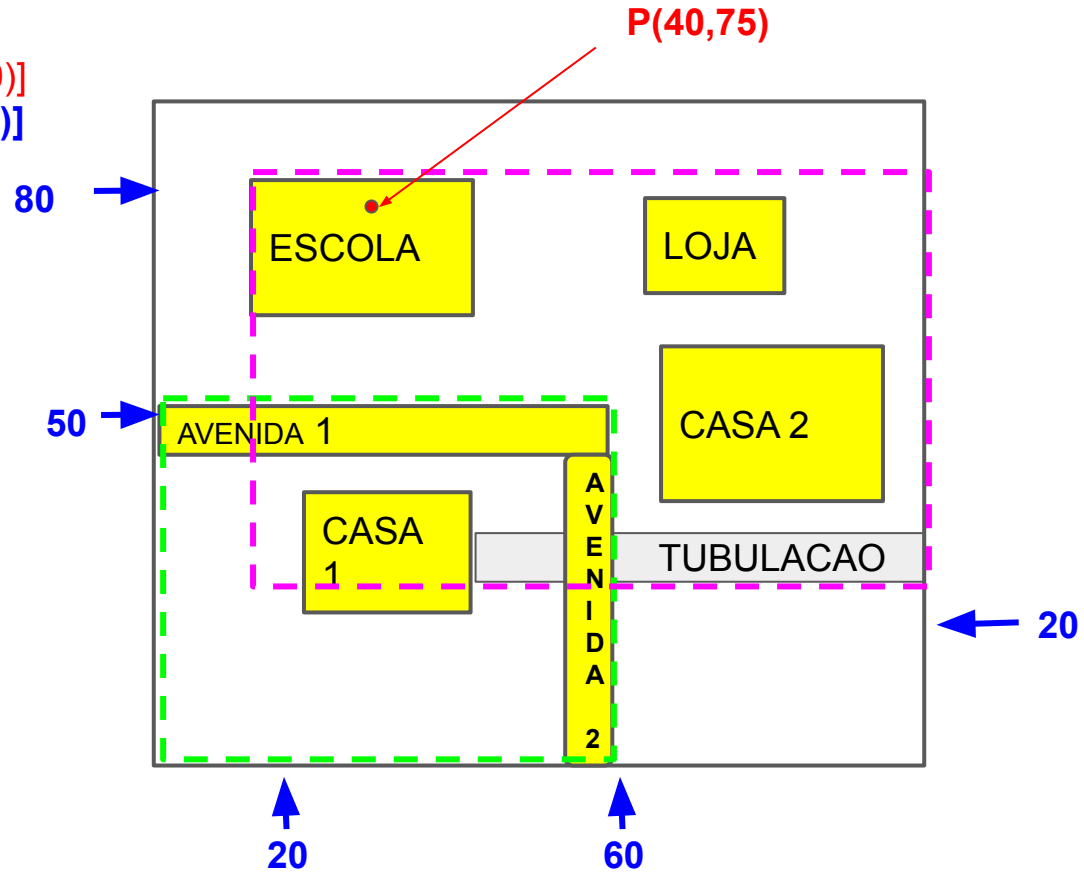




Exemplo: Encontrar objeto que contém o ponto P(40,75)

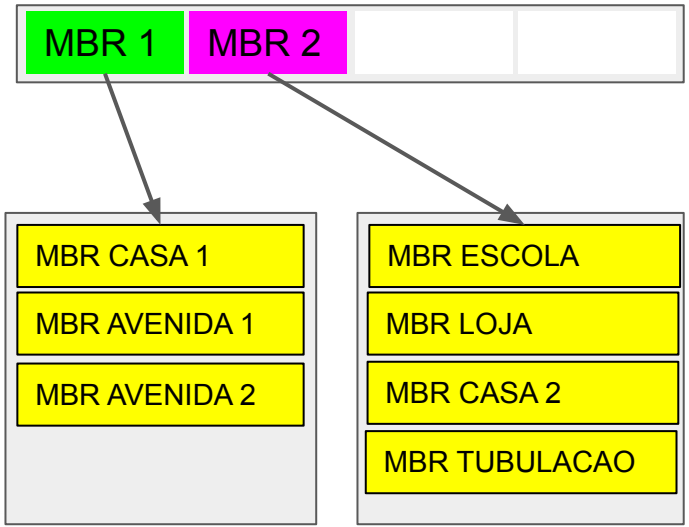
1. P(40,75) **não** está em MBR1 [(0,0),(60,50)]
2. **P(40,75) está em MBR2 [(20,20),(20,100)]**

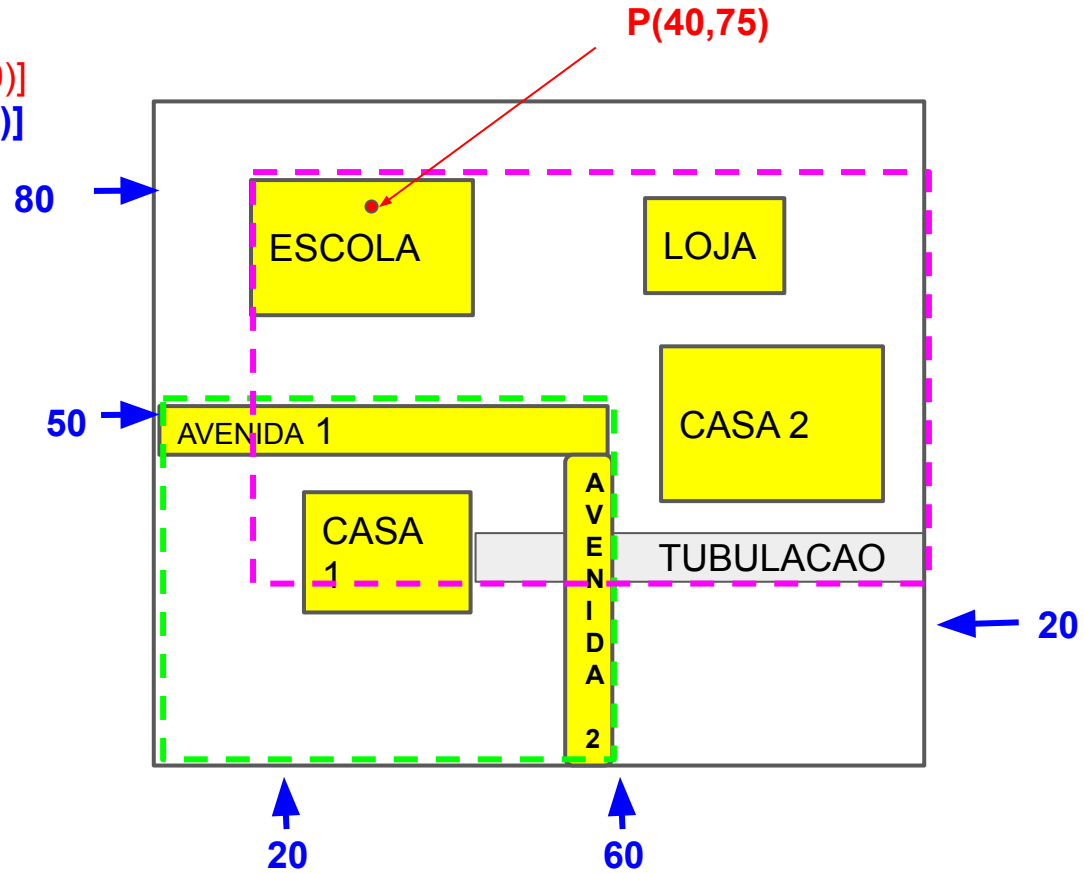




Exemplo: Encontrar objeto que contém o ponto P(40,75)

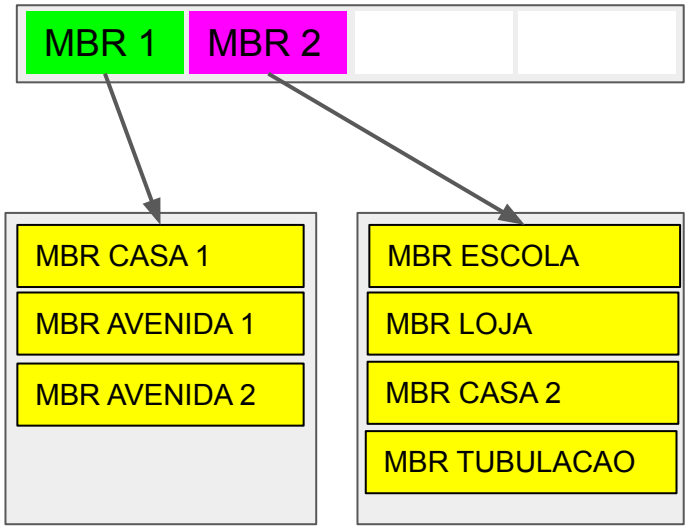
1. P(40,75) **não** está em MBR1 [(0,0),(60,50)]
2. **P(40,75) está em MBR2 [(20,20),(20,100)]**
3. Repetir a busca apenas na segunda subárvore.

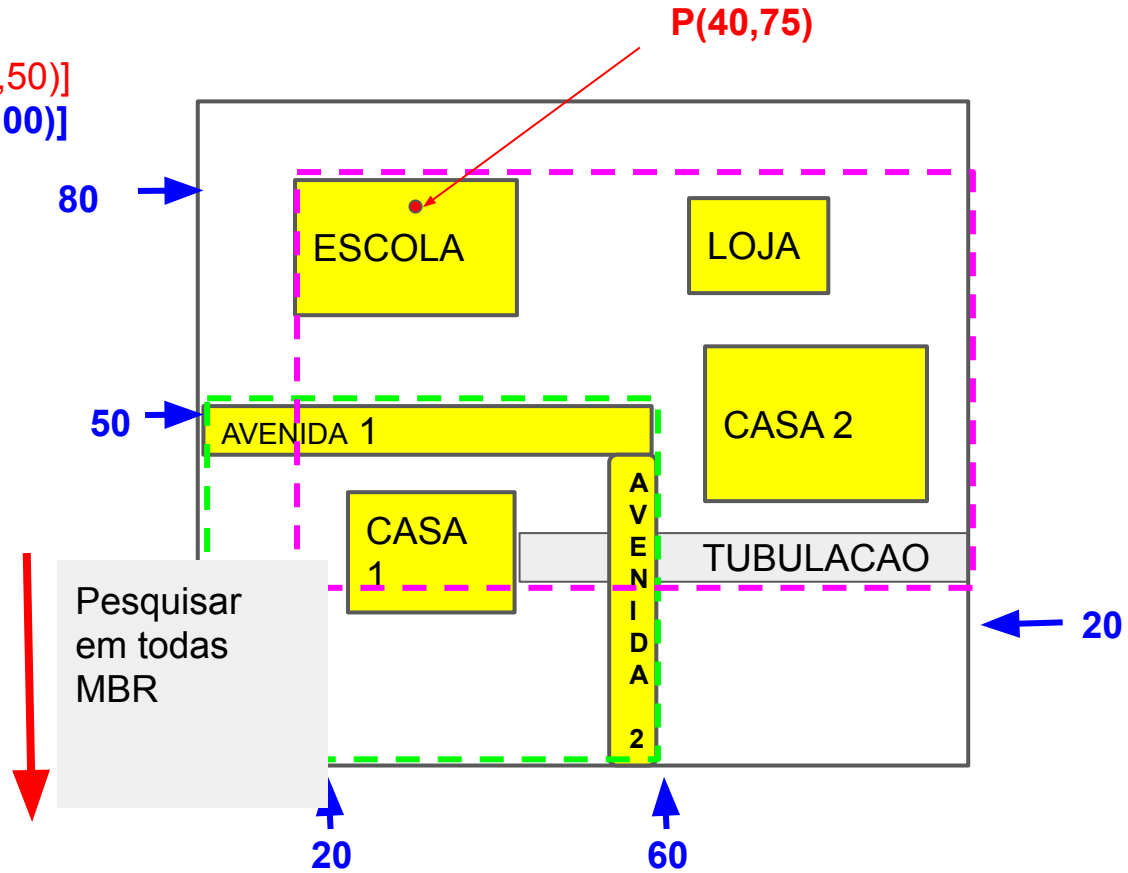




Exemplo: Encontrar objeto que contém o ponto P(40,75)

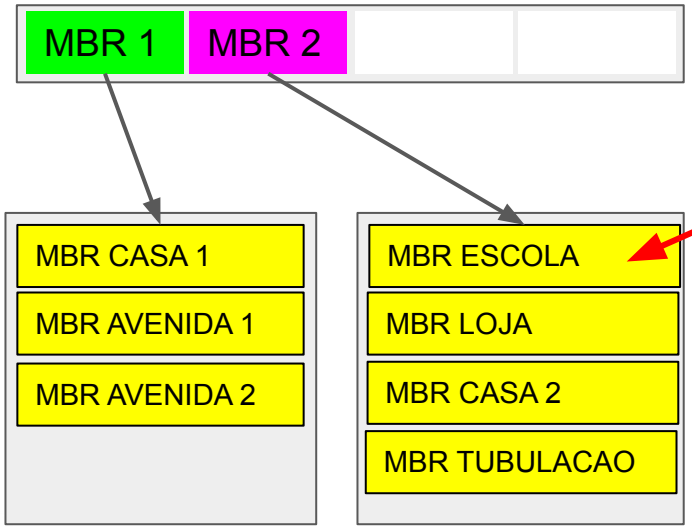
1. P(40,75) **não** está em MBR1 [(0,0),(60,50)]
2. **P(40,75) está em MBR2 [(20,20),(20,100)]**
3. Repetir a busca apenas na segunda subárvore.

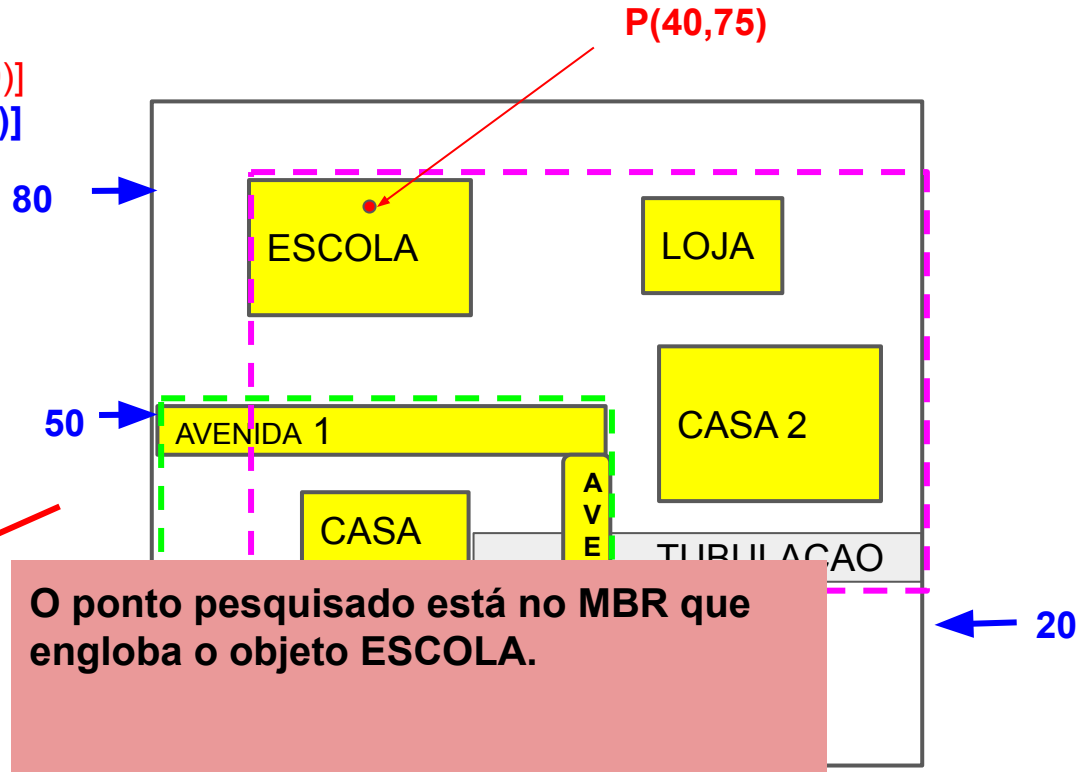




Exemplo: Encontrar objeto que contém o ponto P(40,75)

1. P(40,75) **não** está em MBR1 [(0,0),(60,50)]
2. **P(40,75) está em MBR2 [(20,20),(20,100)]**
3. Repetir a busca apenas na segunda subárvore.

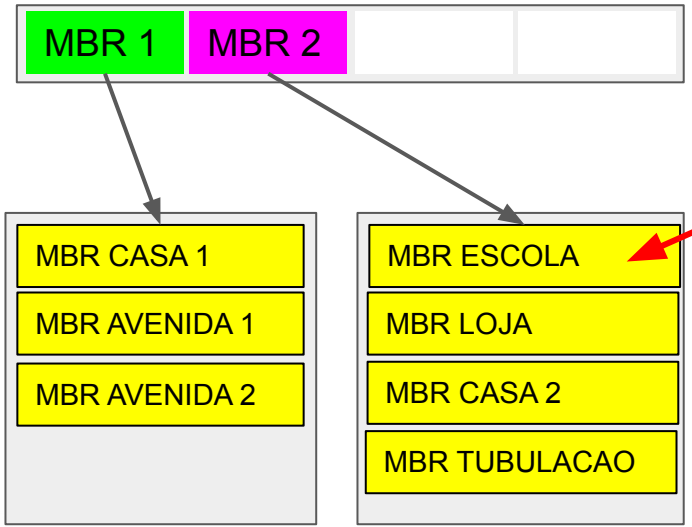


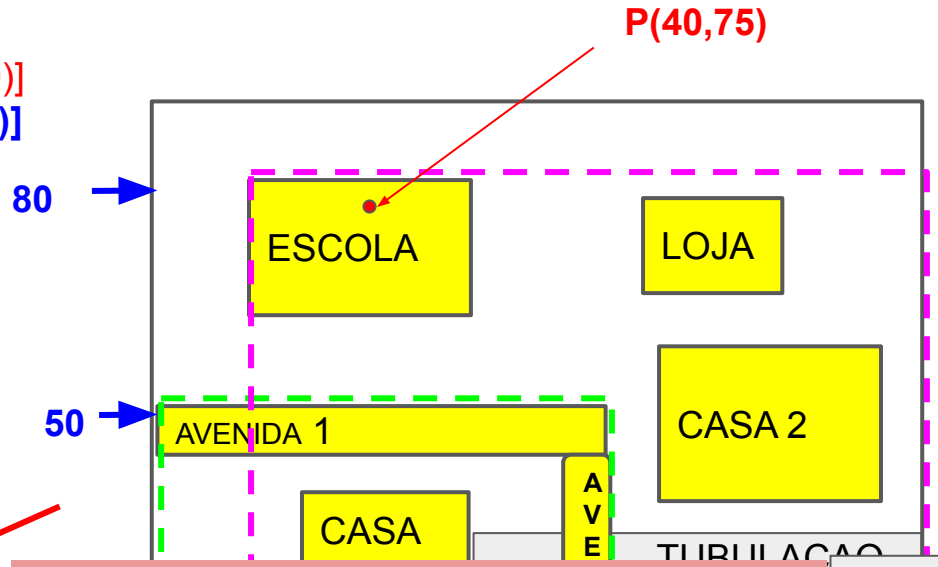


O ponto pesquisado está no MBR que engloba o objeto ESCOLA.

Exemplo: Encontrar objeto que contém o ponto P(40,75)

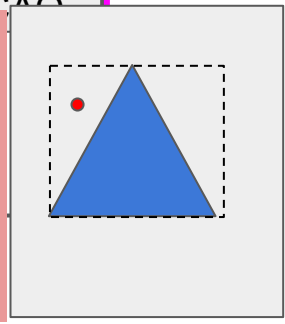
1. P(40,75) **não** está em MBR1 [(0,0),(60,50)]
2. **P(40,75) está em MBR2 [(20,20),(20,100)]**
3. Repetir a busca apenas na segunda subárvore.





O ponto pesquisado está no MBR que engloba o objeto ESCOLA.

IMPORTANTE:
Há a necessidade de pesquisar se o ponto está realmente no objeto.



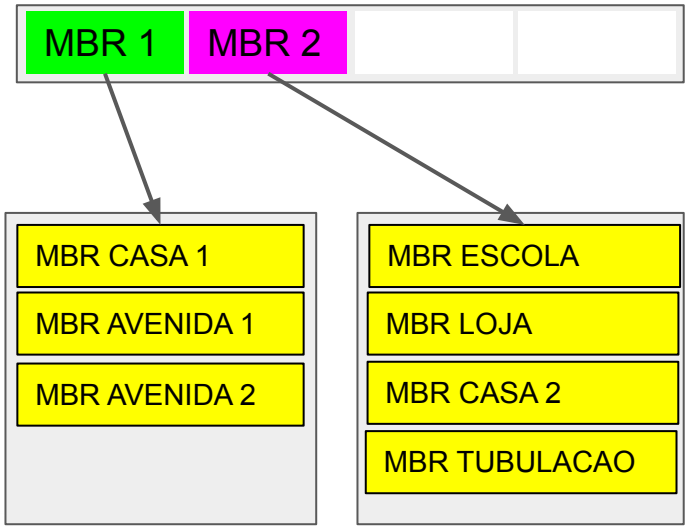
Árvore-R: Inserção

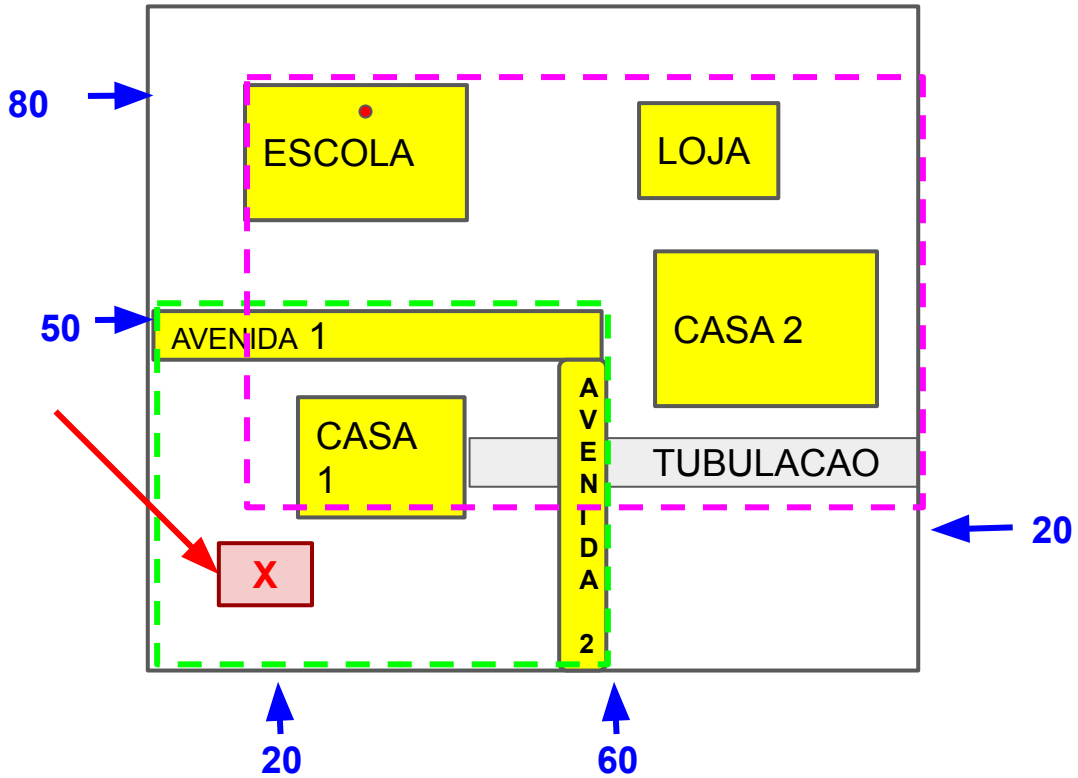
- Uma árvore-R não é única
  - Ela depende da ordem em que os retângulos são inseridos (e possivelmente removidos)
- Algoritmo para inserção de um nó é análogo ao algoritmo de inserção em árvore B
  - Novos retângulos são adicionados a nós folha
  - O nó folha apropriado é determinado pela
    - Navegação na árvore iniciando no nó raiz
    - A cada passo escolhe-se a subárvore cujo retângulo correspondente terá o menor acréscimo de área possível
    - Se ao inserir o nó ocorrer overflow, então executar **cisão**
      - M+1 registros devem ser distribuídos entre dois nós
    - **Cisão** pode propagar até a raiz

Inserção de um objeto na Árvore R

Caso 1:

a. Inserir o objeto **x** na Árvore-R

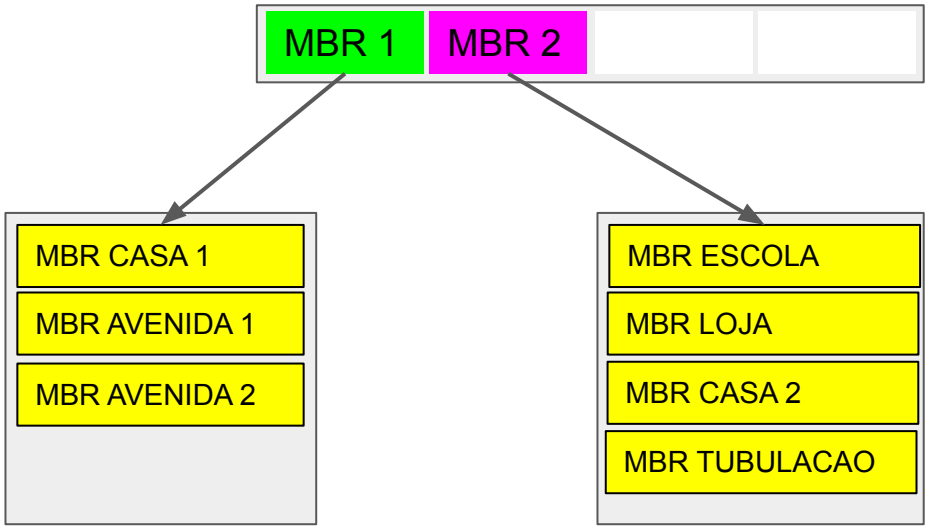




Inserção de um objeto na Árvore R

Caso 1: Inserir o objeto **x** na Árvore-R

x está em está na subárvore apontada por MBR1

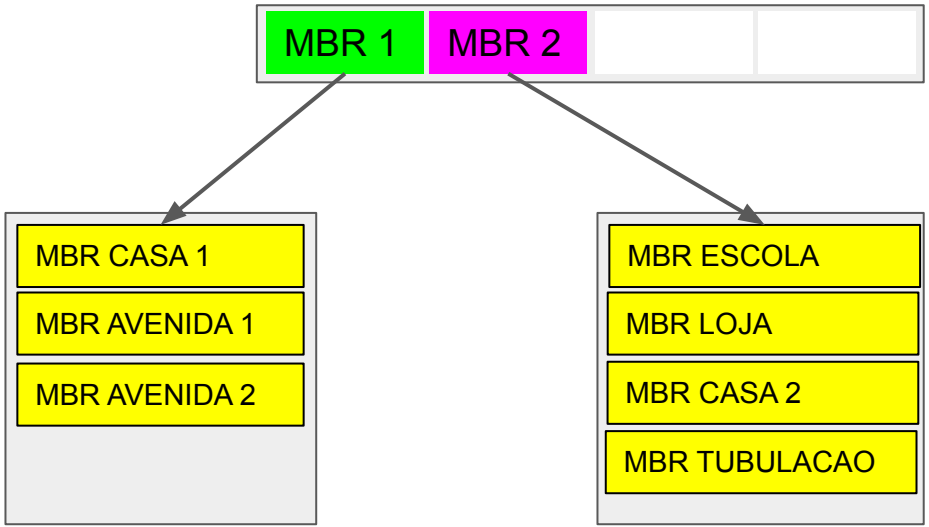


Inserção de um objeto na Árvore R

Caso 1: Inserir o objeto **x** na Árvore-R

x está em está na subárvore apontada por MBR1

Há espaço no nó folha, logo, podemos inserimos o objeto

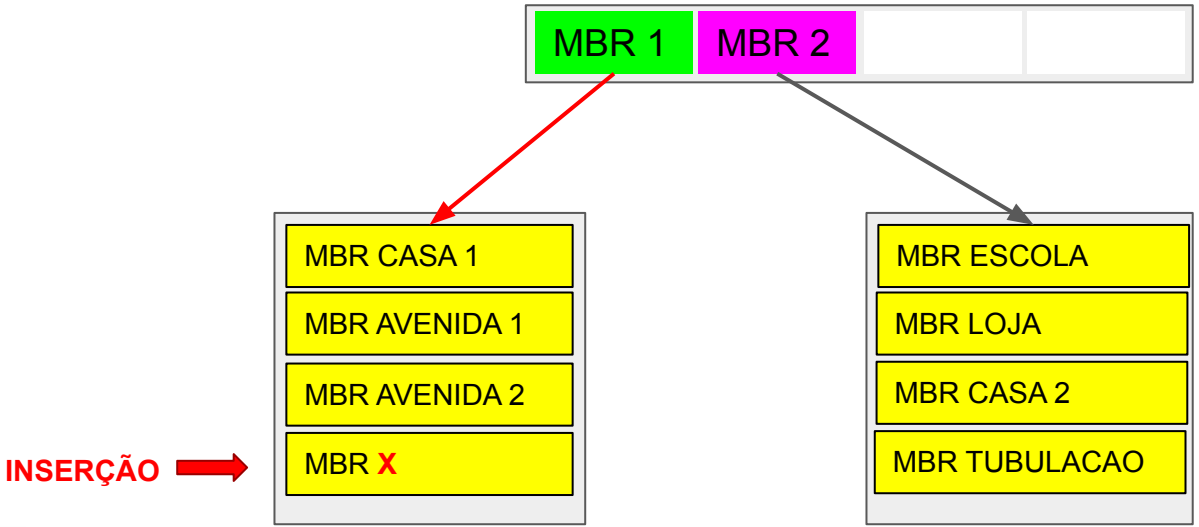


Inserção de um objeto na Árvore R

Caso 1: Inserir o objeto **x** na Árvore-R

x está em está na subárvore apontada por MBR1

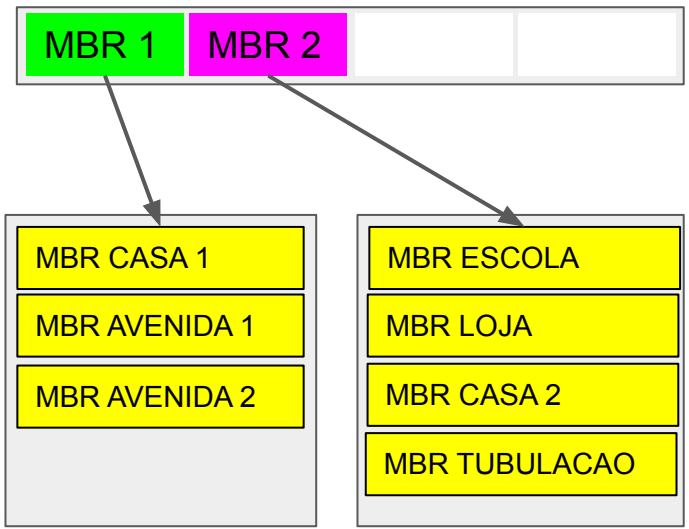
Há espaço no nó folha, logo, podemos inserimos o objeto

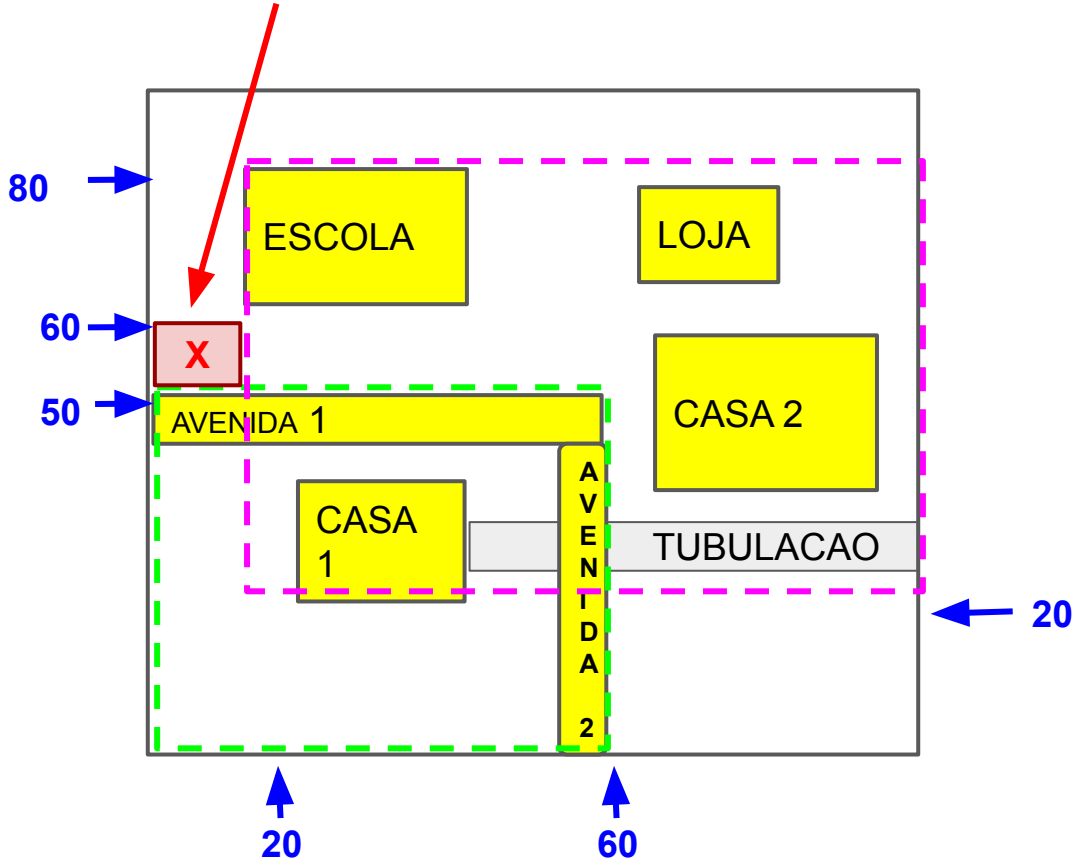


Inserção de um objeto na Árvore R

Caso 2:

a. Inserir o objeto **x** na Árvore-R

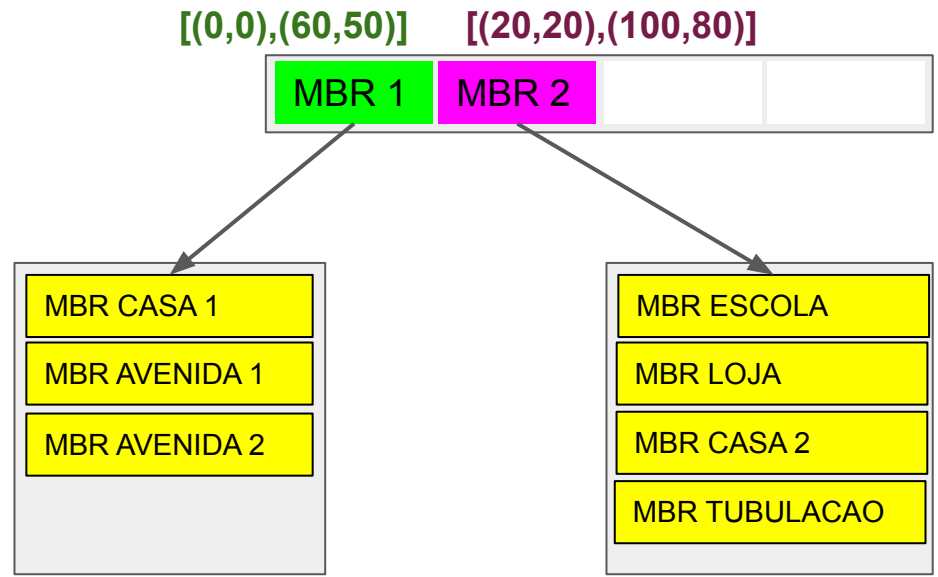




Inserção de um objeto na Árvore R

1. **Exemplo 2:** Inserir o objeto **x** na Árvore-R

Não há nenhum MBR no nó raiz que contenha x

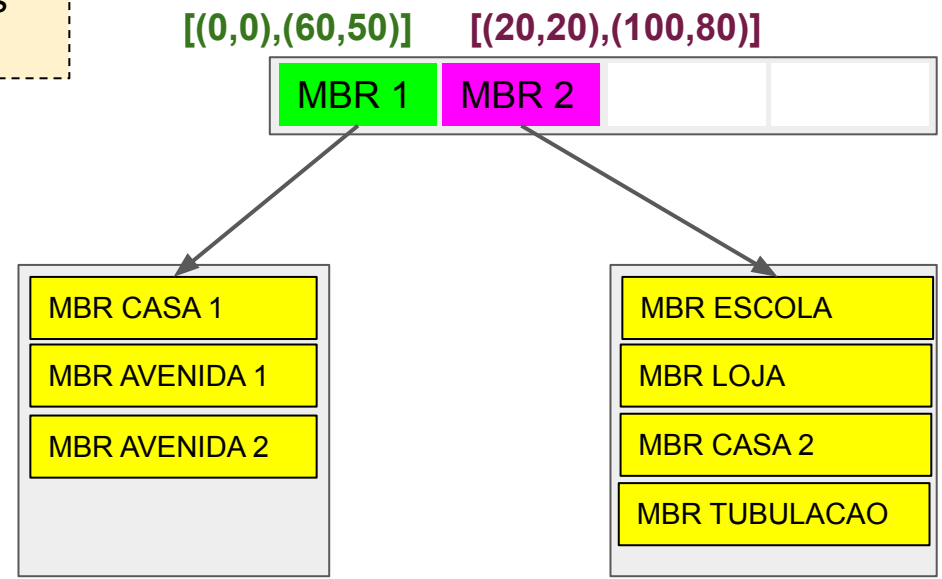


Inserção de um objeto na Árvore R

Caso 2: Inserir o objeto **x** na Árvore-R

Não há nenhum MBR no nó raiz que contenha x

Devemos ajustar os tamanhos de um dos MBRs para incluir x



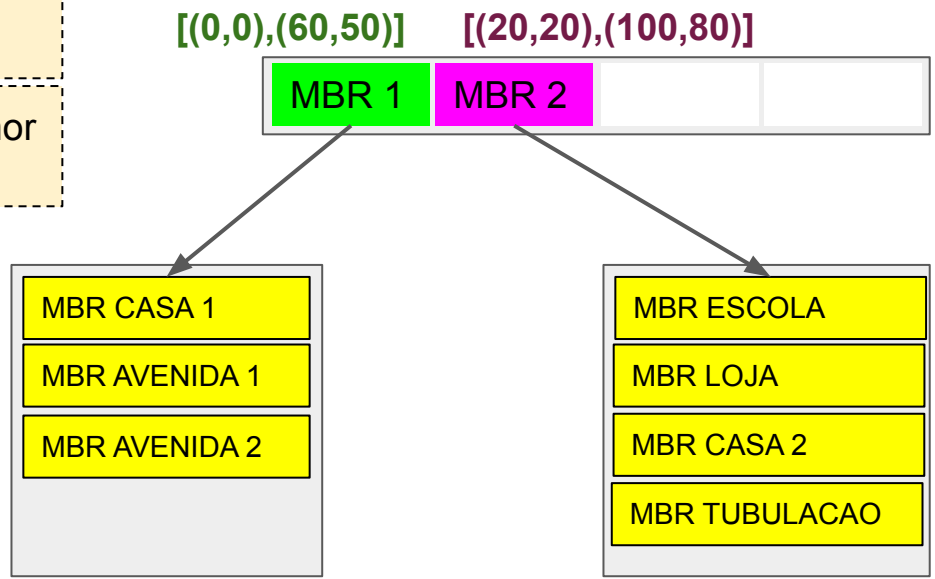
Inserção de um objeto na Árvore R

Caso 2: Inserir o objeto **x** na Árvore-R

Não há nenhum MBR no nó raiz que contenha x

Devemos ajustar os tamanhos de um dos MBRs para incluir x

Amplia o MBR para conter x que adiciona a menor área para a caixa delimitadora.

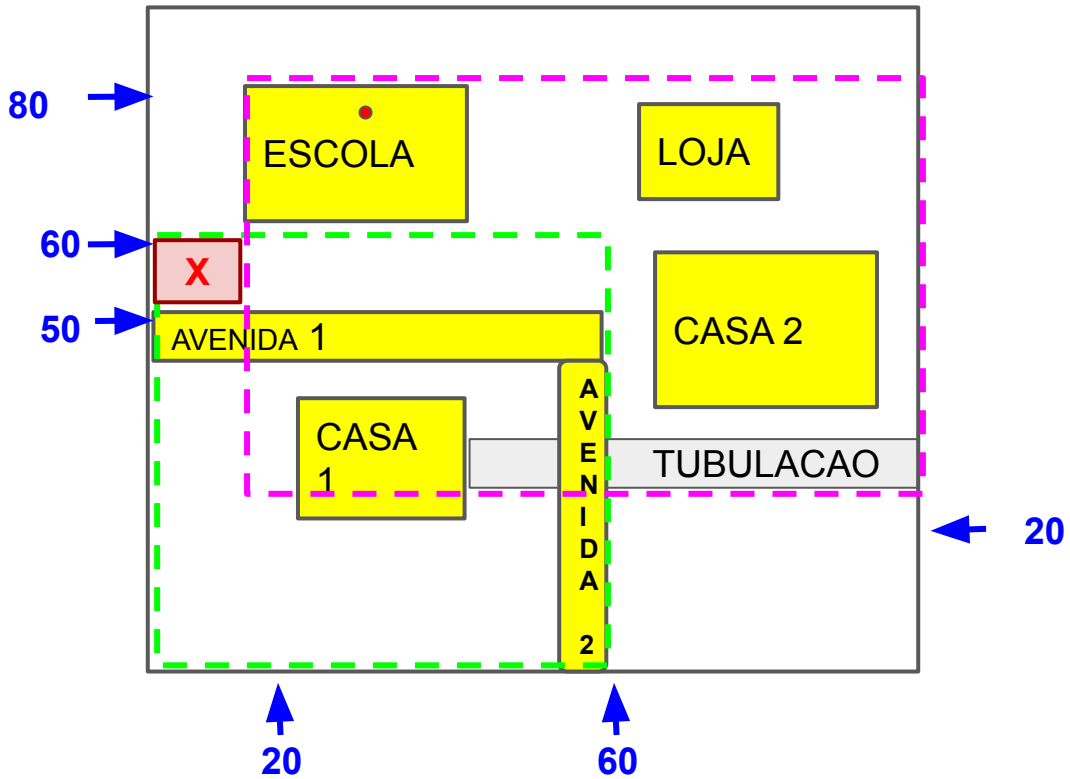


Inserção de um objeto na Árvore R

Caso 2: Inserir o objeto **x** na Árvore-R

Atualize
MBR1

[(0,0),(60,50)] → [(0,0),(60,60)]



Inserção de um objeto na Árvore R

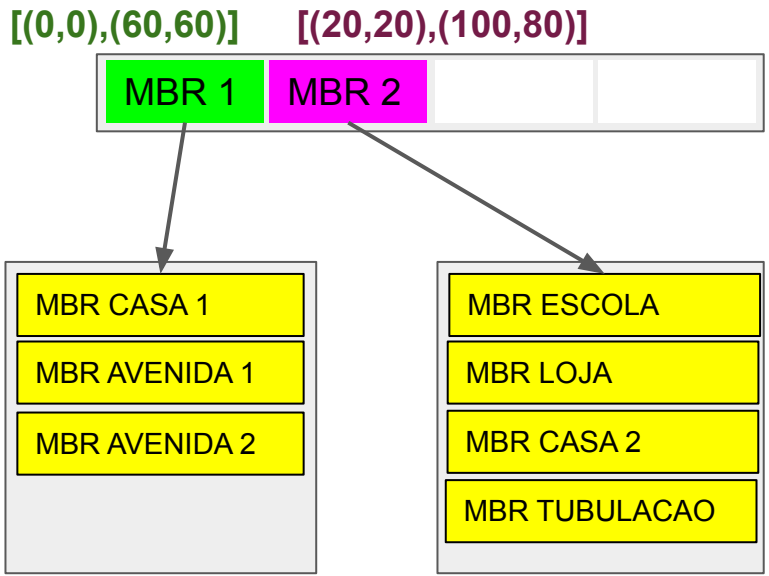
Caso 2: Inserir o objeto **x** na Árvore-R

Não há nenhum MBR no nó raiz que contenha x

Devemos ajustar os tamanhos de um dos MBRs para incluir x

Amplia o MBR para conter x que adiciona a menor área para a caixa delimitadora.

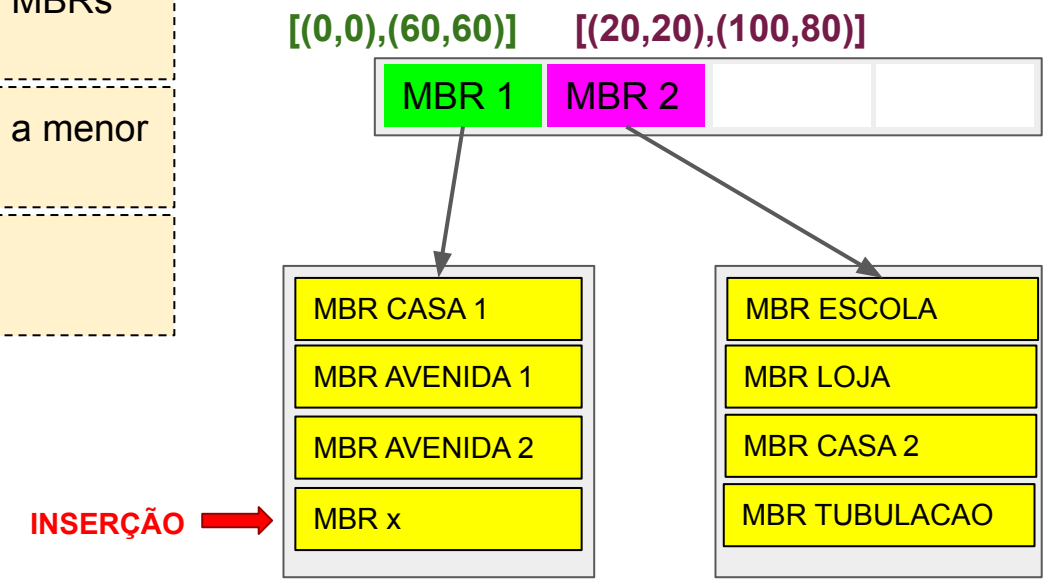
Insere como Caso 1.



Inserção de um objeto na Árvore R

Caso 2: Inserir o objeto **x** na Árvore-R

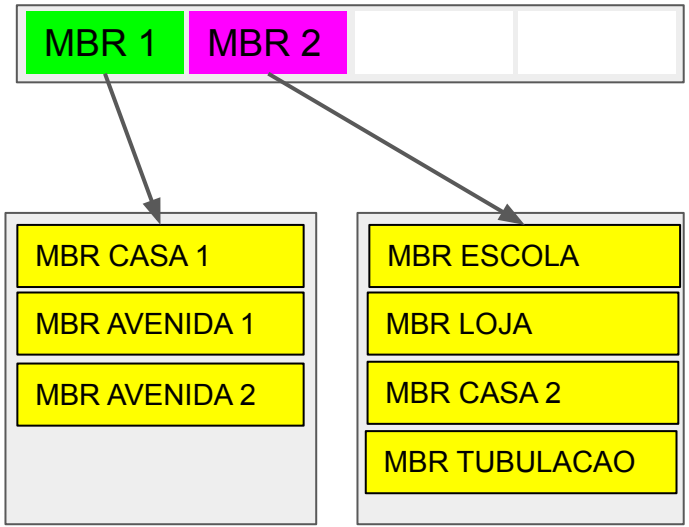
- Não há nenhum MBR no nó raiz que contenha x
- Devemos ajustar os tamanhos de um dos MBRs para incluir x
- Amplia o MBR para conter x que adiciona a menor área para a caixa delimitadora.
- Insere como Caso 1.

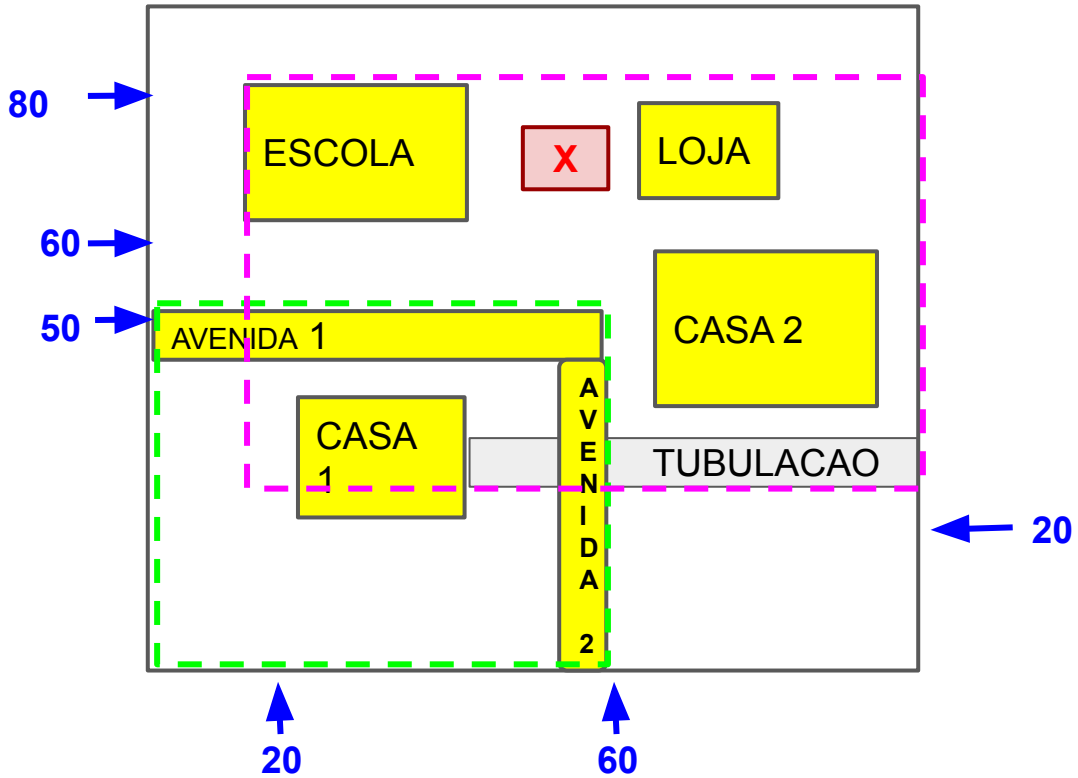


Inserção de um objeto na Árvore R

Caso 3:

a. Inserir o objeto **x** na Árvore-R

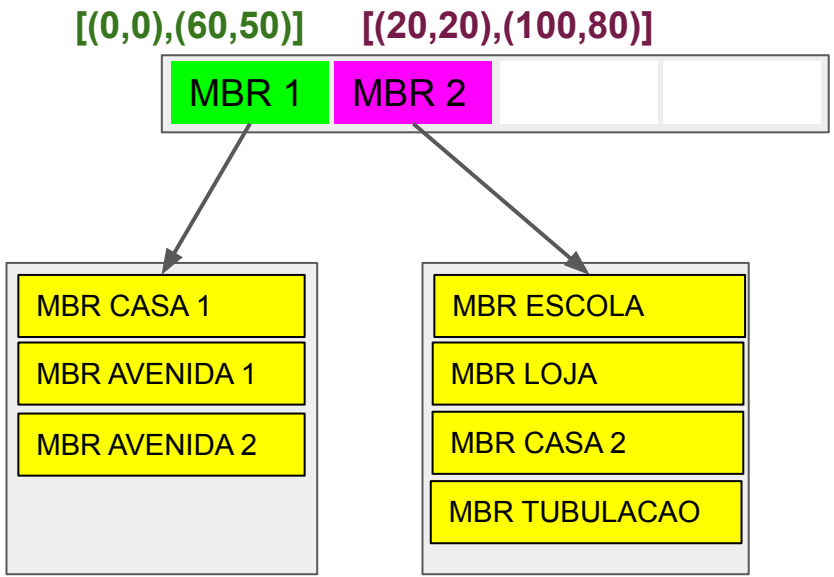




Inserção de um objeto na Árvore R (2,4)

Caso 3: Inserir o objeto **x** na Árvore-R

Objeto X está na subárvore apontada por MBR2

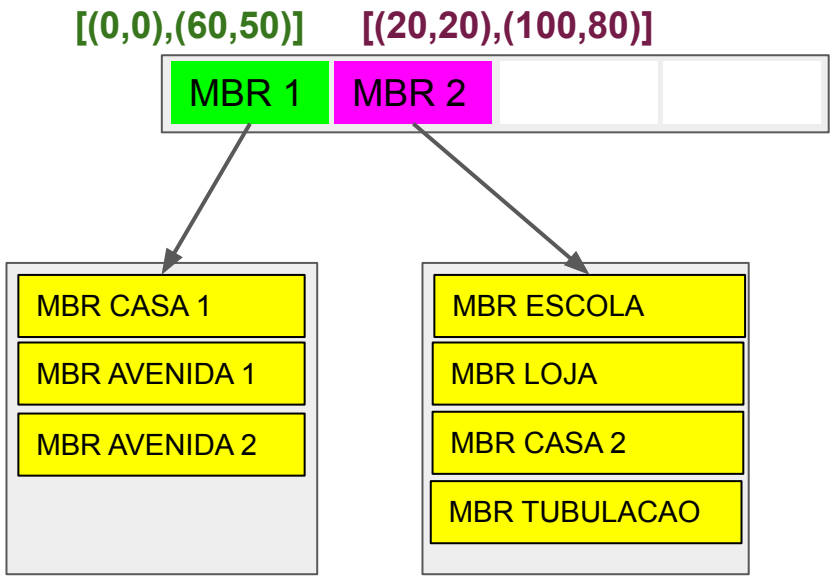


Inserção de um objeto na Árvore R (2,4)

Caso 3: Inserir o objeto **x** na Árvore-R

Objeto X está na subárvore apontada por MBR2

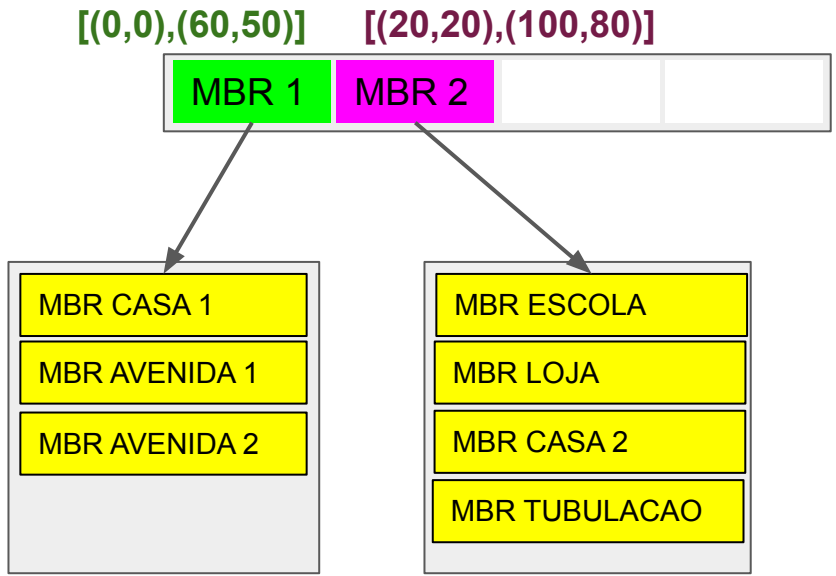
Entretanto, nó folha está completo (4)



Inserção de um objeto na Árvore R (2,4)

Caso 3: Inserir o objeto **x** na Árvore-R

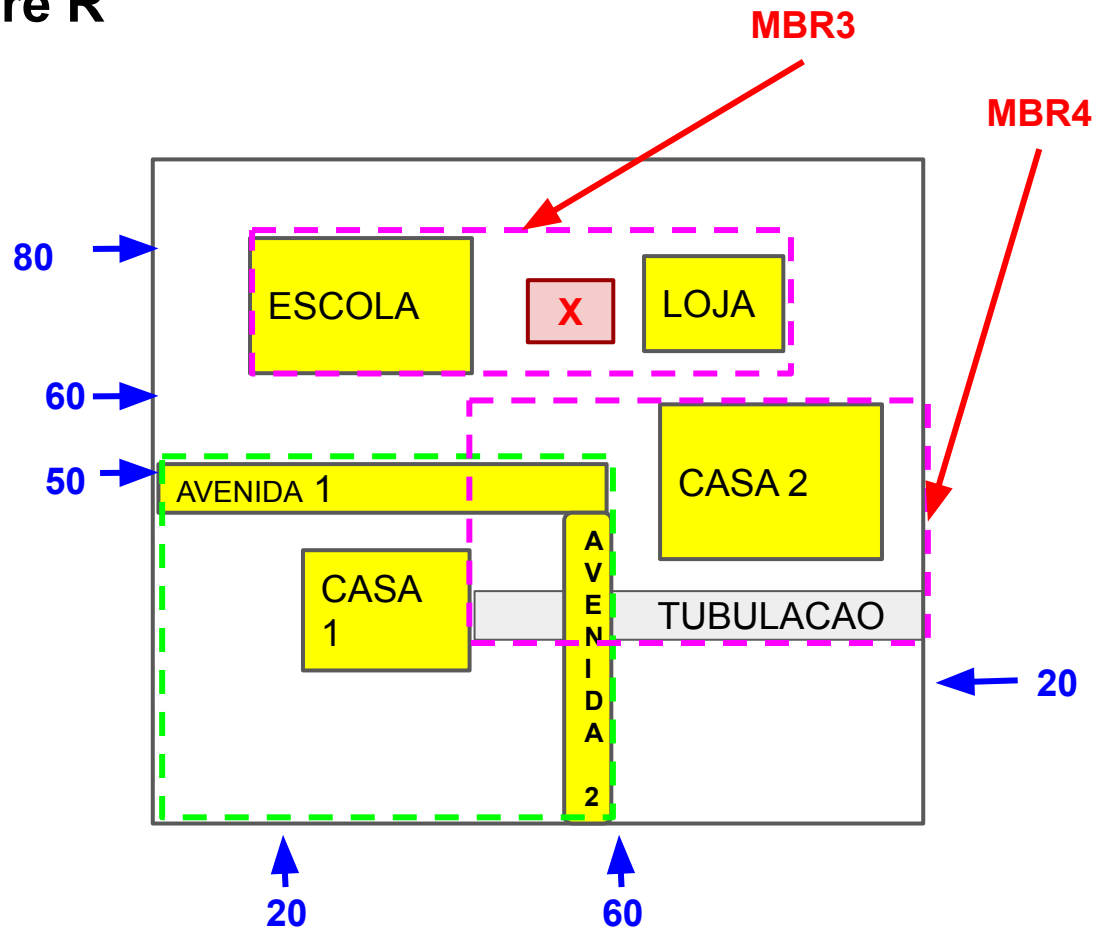
- Objeto X está na subárvore apontada por MBR2
- Entretanto, nó folha está completo (4) (***overflow***)
- Deve-se dividir MBR2 em duas partições



Inserção de um objeto na Árvore R

Caso 3: Inserir o objeto **x** na Árvore-R

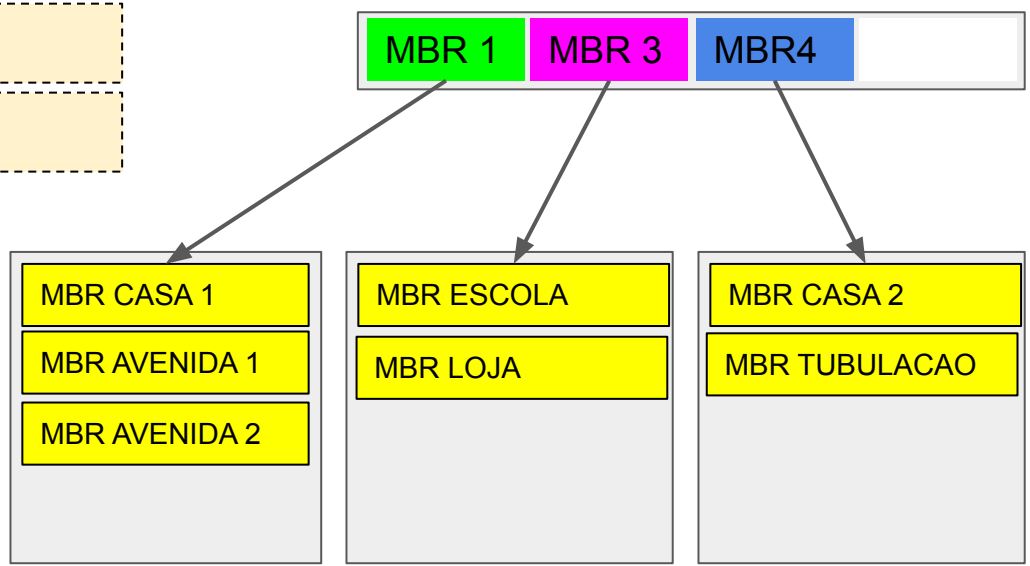
Particionamento de MBR2 em duas partições: MBR3 e MBR4



Inserção de um objeto na Árvore R (2,4)

Caso 3: Inserir o objeto **x** na Árvore-R

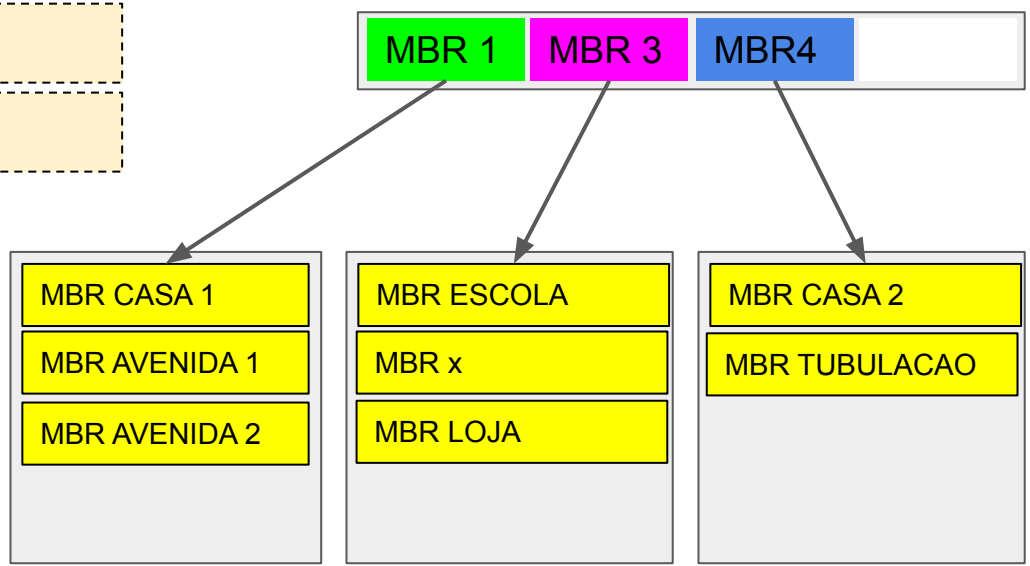
- Objeto X está na subárvore apontada por MBR2
- Entretanto, nó folha está completo (4)
- Deve-se dividir MBR2 em duas partições
- MBR2 é substituído por MBR3 e MBR4



Inserção de um objeto na Árvore R (2,4)

Caso 3: Inserir o objeto **x** na Árvore-R

- Objeto X está na subárvore apontada por MBR2
- Entretanto, nó folha está completo (4)
- Deve-se dividir MBR2 em duas partições
- MBR2 é substituído por MBR3 e MBR4
- Inserir Objeto X como nos casos anteriores



Inserção de um objeto na Árvore R (2,4)

Caso 3: Inserir o objeto **x** na Árvore-R

Objeto X está na subárvore apontada por M

Entretanto, nó folha está completo (4)

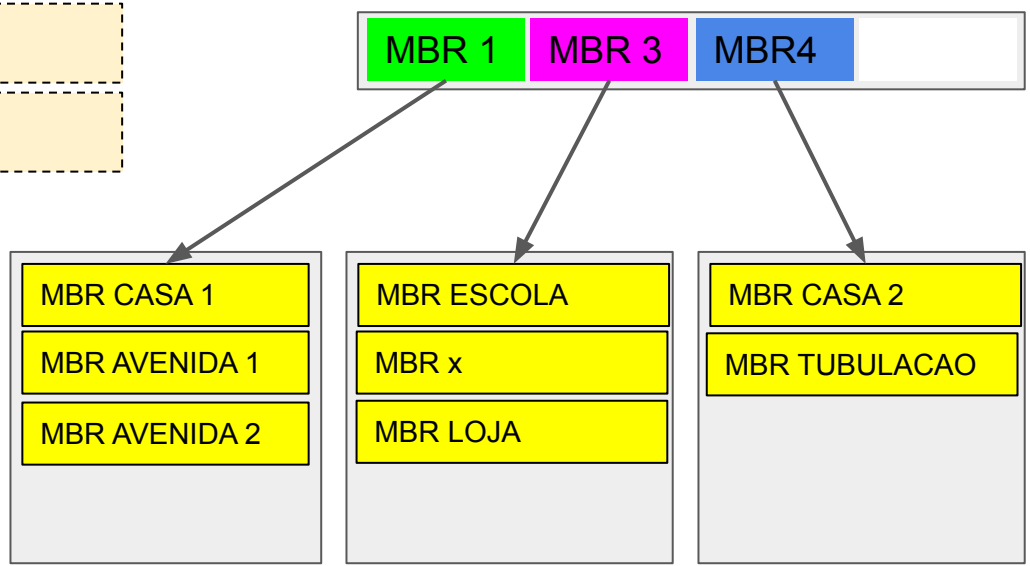
Deve-se dividir MBR2 em duas partições

MBR2 é substituído por MBR3 e MBR4

Inserir Objeto X como nos casos anteriores

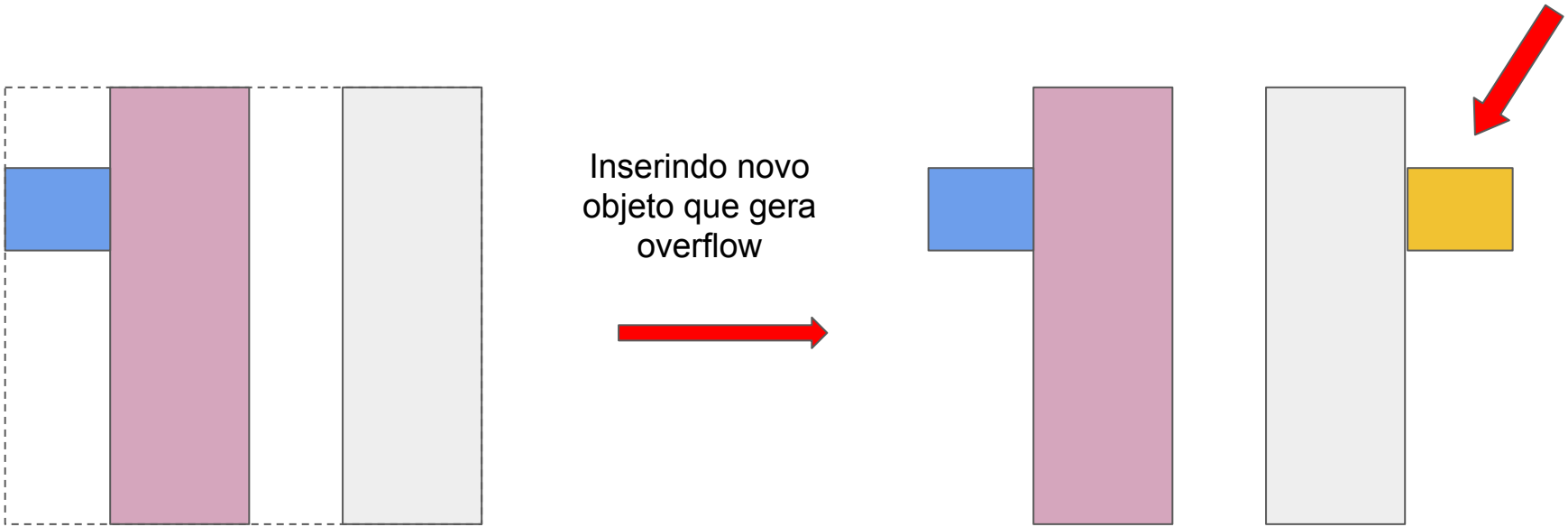
Obs: Insere ao invés de manter MBR 2. Motivos:

- Mais simples de implementar pois aproveita inserção.
- reinserções refinam a estrutura espacial.



Particionamento de Objetos

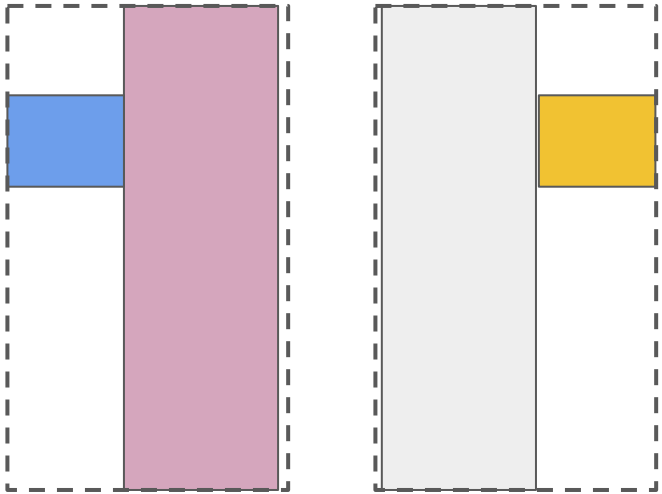
Considere um nó folha de uma árvore R contendo os seguintes objetos:



Como dividir objetos em dois grupos?

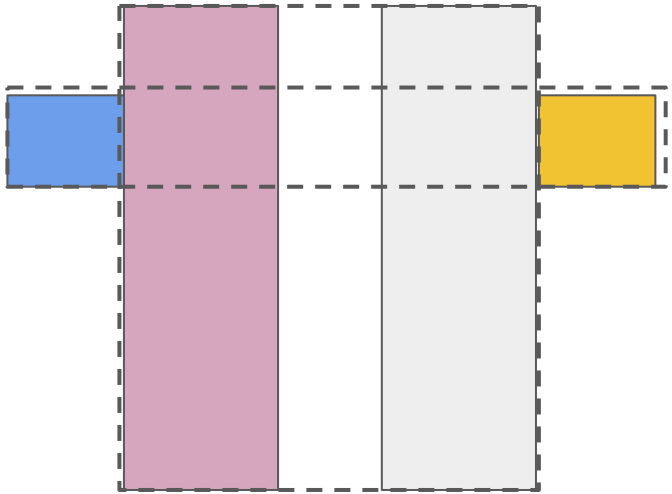
Particionamento de Objetos

Comparando particionamentos



Ineficiente

A área total da caixa delimitadora é menor



Eficiente

Objetivo ao particionar é minimizar as áreas.

Particionamento de Objetos

- Na definição de R-Tree existem 3 possíveis algoritmos para particionamento:
  - **exaustivo** - gera todas as divisões possíveis e escolhe a melhor (menor área). Leva muito tempo, as outras opções são mais práticas.
  - **custo quadrático** - escolhe duas entradas das M+1 que gastam mais área se fossem colocadas no mesmo nó e essas se tornam as primeiras entradas de cada novo nó. As demais entradas são colocadas em cada grupo calculando o aumento de área necessário em cada escolha.
  - **custo linear** - parecido com quadrático mas busca os retângulos mais extremos nas dimensões e escolhe o par mais extremo para formar os primeiros integrantes dos nós (menor lado seja o maior entre as opções)

Remoção

- Remover retângulo R de uma árvore R
  - Localizar nó folha L que contém R e remover R de L
  - Ajustar os MBRs no caminho de L até a raiz
    - Todos os nós onde ocorrer *underflow* são armazenados em um conjunto U
  - Quando a raiz é alcançada
    - Se ela tem apenas um único filho, então o filho se torna a nova raiz
  - Os nós onde ocorreu *underflow* são reinseridos na árvore
    - Entradas de U que correspondem a nós folha são incluídas em nós folha
    - Outros nós, têm suas entradas posicionadas no nível que faça com que seus nós folha continuem no mesmo nível dos demais.

R-Tree - Variações

Da mesma forma que a árvore B teve variações desde que foi proposta, a árvore R também já teve propostas de melhorias:

- R+-Tree - foca em retângulos menores e evita que várias sub-árvores precisem ser pesquisadas.
- R*-Tree - publicada em 1990 - foca em minimizar área dos retângulos e minimiza também as sobreposições. E maximiza a capacidade de armazenamento. É a que possui melhor desempenho.

Referências:

- Notas de Aulas do Prof. Professor – Jairo Francisco de Souza http://www.ufjf.br/jairo_souza/files/2012/11/5-Indexa%C3%A7%C3%A3o-Arvore_R.pdf
- IBM Knowledge Center - R-Trees index structure https://www.ibm.com/support/knowledgecenter/en/SSGU8G_12.1.0/com.ibm.rtree.doc/ids_rti_007.htm
- Cheung's Home Page, http://www.mathcs.emory.edu/~cheung/Courses/554/Syllabus/3-index/R-tree3.html
- Guttman, R-Trees: A Dynamic Index Structure for Spatial Searching, SIGMOD 1984.
- Beckmann, Kriegel, Schneider, Seeger, The R*-tree: an efficient and robust access method for points and rectangles, SIGMOD 1990